第 1 章　C 语言概述及程序设计基础

- 了解 C 语言的发展和基本特点，掌握 C 语言程序的基本结构
- 了解 C 程序编译预处理的宏定义、条件编译，掌握文件包含的使用方法
- 掌握程序设计算法的基本概念和算法描述的基本工具，学会运用传统流程图描述一个具体的算法
- 熟悉 C 语言编程环境 Visual C++ 6.0 的使用，掌握程序的书写风格

1.1　C 语言的发展及特点

1.1.1　C 语言的发展

C 语言是国际上流行的计算机高级程序设计语言。与其他高级语言相比，C 语言的硬件控制能力和运算表达能力强，可移植性好，效率高。所以，C 语言仍然是当今最流行、最受欢迎的计算机语言之一，应用面非常广，许多大型软件都使用 C 语言编写。

C 语言起源于一种面向问题的高级语言——ALGOL60 语言。1963 年英国剑桥大学推出 CPL 语言，此语言在 ALGOL 语言的基础上增加了硬件处理能力，同年剑桥大学的马丁·理查德对其简化，提出 BCPL 语言；1970 年美国贝尔实验室的肯·汤姆逊进一步简化，提出了 B 语言（取 BCPL 的第一个字母）；1972 年美国贝尔实验室的布朗·W·卡尼汉和丹尼斯·M·利奇对其完善和扩充，提出了 C 语言（取 BCPL 的第二个字母）；1987 年美国标准化协会制定了 C 语言标准“ANSI C”，即现在流行的 C 语言。

用 C 语言开发的系统非常多，例如 UNIX、dBASE 以及 Windows 和 Office 的核心程序等。

本书以 Visual C++ 6.0（后文简称为 VC）为学习平台。

1.1.2　C 语言的特点

C 语言具有以下基本特点：

（1）C 语言是具有低级语言功能的高级语言。C 语言既具有高级语言的功能，又具有低级语言的许多功能。它把高级语言的基本结构和语句与低级语言的实用性结合起来，是处于汇编语言和高级语言之间的一种程序设计语言，也可称其为“中级语言”。

（2）C 语言简洁、紧凑，使用方便、灵活。C 语言一共只有 32 个关键词、9 种控制语句。C 程序书写形式自由，主要用小写字母表示，相对于其他高级语言而言源程序短。

（3）运算符丰富，表达式能力强。C 语言共有 34 种运算符，范围广泛，除一般高级语言所使用的算术、关系和逻辑运算符外，还可以实现以二进制位为单位的运算，并且具有如 a++、--b 等单目运算符和+=、-=、*=、/=等复合运算符等。

（4）数据结构丰富，便于数据的描述与存储。C 语言具有丰富的数据结构，其数据类型有整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等，因此能实现复杂的数据结构的运算。

（5）C 语言是结构化、模块化的编程语言。程序的逻辑结构可以使用顺序、分支和循环 3 种基本结构组成。C 语言程序采用函数结构，十分便于把整体程序分割成若干相对独立的功能模块，并且为程序模块间的相互调用以及数据传递提供了便利。

（6）可使用宏定义。C 语言程序中，可使用宏定义编译预处理语句和条件编译预处理语句，为编程提供了方便。

（7）可移植性好。与汇编语言相比，C 程序基本上不作修改就可以运行于各种型号的计算机和各种操作系统。

C 语言也存在一些不足之处，如运算符及其优先级过多、语法定义不严格等，对于初学者来说有一定的困难。

由于 C 语言具有上述特点，因此 C 语言得到了迅速推广，成为人们编写大型软件的首选语言之一。许多原来用汇编语言处理的问题可以用 C 语言来处理了。

1.2 C 语言程序的基本结构

1.2.1 C 语言程序的基本结构

我们先通过两个简单的例子来了解一下 C 程序。

【例 1-1】一个简单的 C 程序示例。（ex1_1.cpp）

```
#include <stdio.h>                              //预处理，包含文件

void main()                                     //主函数
{
    printf("这是我的第一个 C 语言程序!\n");     //调用输出函数
}
```

程序的运行结果如图 1-1 所示。

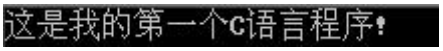


图 1-1　例 1-1 的运行结果

在 VC 中运行程序后，显示结果的窗口中，可以看到提示信息“Press any key to continue”，告诉用户可以按任意键关闭显示结果的窗口。

【例 1-2】已知圆的半径，求圆的周长和面积。（ex1_2.cpp）

```
#include <stdio.h>
#define PI 3.14159                              //预处理，宏定义

float area(int r)                               //计算半径为 r 的圆面积函数
{
    return PI * r * r;                          //返回圆面积
}
void main()
{
```

```
    int r;                                  //声明圆半径 r 为整型变量
    float l,s;                              //声明周长 l、面积 s 为单精度浮点型变量
    scanf("%d",&r);                         //从键盘输入半径
    l=2 * PI * r;                           //计算周长 l 的值
    s=area(r);                              //计算面积 s 的值
    printf("r=%d,l=%5.2f,s=%5.2f\n",r,l,s); //输出圆的半径、周长和面积
}
```

程序的运行结果如图 1-2 所示。

图 1-2　例 1-2 的运行结果

注意：在图 1-2 中，第一行的 5 是在程序运行后从键盘输入的，输入 5 之后，然后按回车键；第二行是程序运行后显示的结果。

例 1-2 由主函数 main 和被调用函数 area 组成，在主函数中输入半径 r，然后通过语句 s=area(r) 调用函数 area，计算结果由 return 语句返回给主函数。这两个函数在位置上是独立的，可以把主函数 main 放在前面，也可以把主函数 main 放在函数 area 的后面。

scanf 和 printf 是 C 语言提供的标准输入/输出函数，&r 中的“&”的含义是取地址，程序中 scanf 函数的作用是将从键盘上键入的一个数输入到变量 r 所标志的内存单元中，或者称对 r 赋值。

从以上两个例子中可以看到，一个 C 语言程序的基本结构分为两个部分：

1. 预处理部分

#include <stdio.h> 是一条预处理命令，用“#”号开头，后面不能加“;”号，stdio.h 是系统提供的头文件，其中包含有关输入/输出函数的信息，如程序中用到的 printf 输出函数。输出语句中的“%d,%5.2f”为输出格式符，它指定输出结果时的数据类型和格式，程序在执行时，该位置由具体数据替代。更详细的使用方法在第 2 章中进行说明。

2. 函数部分

函数是 C 语言程序的基本单位，用于描述程序所完成的功能。

程序中 main 是主函数名，C 语言规定必须用 main 作为主函数名，函数名后的一对圆括号不能省略，圆括号中内容（参数）可以是空的。

一个 C 程序可以包含任意多个函数，但必须有且只有一个 main 主函数。一个 C 程序总是从主函数开始执行，最后在主函数结束。主函数的位置在程序中是任意的，其他函数总是通过函数调用语句来执行。

主函数可以调用任何其他函数，任何非主函数之间也可以相互调用，但是均不能调用主函数。

函数包含两部分：①函数的首部，包括函数类型、函数名和形式参数；②函数体，需要用花括号括起来，左括号表示函数体的开始，右括号表示函数体的结束。其函数定义形式可表示如下：

```
函数类型 函数名（形式参数列表)
{
    函数体
}
```

主函数的类型为 void，表示是空类型，C 语言规定：空类型的函数不需要返回值。函数体

中包含各种语句，语句是程序的基本执行单位，语句可分为定义（声明）部分和执行语句部分。每一条语句都必须用分号“;”结束，语句的数量不限，程序中由这些语句向计算机系统发出指令，如程序中 printf 输出语句。

C 语言本身没有输入/输出语句。输入和输出操作是由调用系统提供的输入/输出函数来完成的。

程序中为了对程序代码进行说明，可以添加注释，作用是帮助用户阅读程序，它对程序的运行不起作用，在对源程序进行编译时，注释会被忽略。注释内容可以是西文，也可以是中文，注释通常用于说明变量的含义、程序段的功能。注释部分可以放在程序中任意合适的位置，一个好的程序应该有必要的注释，这样可以增加可读性。C 程序中的注释有如下两种：

（1）/*……*/表示注释一个程序块，注释内容可以是一行或多行。“/*”和“*/”必须成对出现，且“/”和“*”之间不能有空格。

（2）//用于注释一行，在一行中“//”后面的内容将都被注释。

本书中程序文件的命名采用的格式为 ex1_1.cpp、ex1_2.cpp 等，其中下划线前面的数字表示章节序号，下划线后面的数字表示该示例在章节中的顺序号。统一的命名易于查找和比较。

1.2.2 C 语言的关键字

C 语言中的关键字（也称保留字），由 ANSI 标准定义的 C 语言关键字共 32 个，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

所有的关键字都有固定的意义，不能用作其他。

所有的关键字都必须小写，如：else 与 ELSE 代表不同的含义，else 是关键字，ELSE 用户定义的标识符。

根据关键字的作用，可分为数据类型关键字、控制语句关键字、存储类型关键字和其他关键字 4 类。

1. 数据类型关键字（12 个）

（1）char：声明字符型变量或函数

（2）double：声明双精度变量或函数

（3）enum：声明枚举类型

（4）float：声明浮点型变量或函数

（5）int：声明整型变量或函数

（6）long：声明长整型变量或函数

（7）short：声明短整型变量或函数

（8）signed：声明有符号类型变量或函数

（9）struct：声明结构体变量或函数

（10）union：声明联合数据类型

（11）unsigned：声明无符号类型变量或函数

（12）void ：声明函数无返回值或无参数，声明无类型指针

2. 控制语句关键字（12 个）

1）循环语句

（1）for：一种循环语句

（2）do：循环语句的循环体

（3）while：循环语句的循环条件

（4）break：结束本层循环

（5）continue：结束本次循环，开始下一次循环

2）条件语句

（1）if：条件语句

（2）else：条件语句否定分支（与 if 连用）

（3）goto：无条件跳转语句

（4）switch：用于多分支语句

（5）case：多分支语句分支

（6）default：多分支语句中的“其他”分支

3）子程序返回语句 return

3. 存储类型关键字（4 个）

（1）auto：声明自动变量（一般不使用）

（2）extern：声明变量是在其他文件中声明（也可以看做是引用变量）

（3）register：声明寄存器变量

（4）static：声明静态变量

4. 其他关键字（4 个）

（1）const：声明只读变量

（2）sizeof：计算数据类型长度

（3）typedef：用以给数据类型取别名（当然还有其他作用）

（4）volatile：说明变量在程序执行中可被隐含地改变

1.3　编译预处理

在 1.2 节中，用过以“#”号开头的预处理命令，如包含命令#include、宏定义命令#define 等。在源程序中这些命令都放在函数之外，而且一般都放在源文件的开头，它们称为预处理部分。

所谓预处理是指C语言编译器在对源程序进行编译过程中的第一遍扫描之前所作的工作。预处理是 C 语言的一个重要功能，它由预处理程序负责完成。当对一个源文件进行编译时，系统将自动引用预处理程序对源程序中的预处理部分作处理，处理完毕后，自动进入对源程序的编译（词法扫描和语法分析），最后生成目标代码。

C 语言提供了多种预处理功能，如宏定义、文件包含、条件编译等。合理地使用预处理功能编写的程序便于阅读、修改、移植和调试，也有利于模块化程序设计。本节将介绍常用的几种预处理功能。

1.3.1　宏定义

在 C 语言源程序中允许用一个标识符来表示一个字符串，称为“宏”。被定义为“宏”的

标识符称为“宏名”。在编译预处理时，对程序中所有出现的“宏名”，都用宏定义中的字符串来代换，这称为“宏代换”或“宏展开”。

宏定义是由源程序中的宏定义命令完成的。宏代换是由预处理程序自动完成的。

在 C 语言中，宏分为有参数和无参数两种。下面分别讨论这两种宏的定义和调用。

1. 无参宏定义

无参宏的宏名后不带参数。其定义的一般形式为：

```
#define  标识符  字符串
```

其中的“#”表示这是一条预处理命令。凡是以“#”开头的均为预处理命令。define 为宏定义命令。“标识符”为所定义的宏名。“字符串”可以是常数、表达式、格式串等。

对符号常量的定义就是一种无参宏定义。此外，常对程序中反复使用的表达式进行宏定义。

【例 1-3】无参宏定义示例程序。（ex1_3.cpp）

```
#define X2 x*x
#define M (y*y+2*y)
main(){
    int x,y;
    x=3;
    y=5;
    printf("%d\n",X2);
    printf("%d\n",M*2);
}
```

程序运行结果如图 1-3 所示。

图 1-3　例 1-3 运行结果

上例程序中首先进行宏定义，定义了在源代码中用 M 来替代表达式(y*y+2*y)，在printf("%d\n",M*2)中作了宏调用。在预处理时经宏展开后该语句变为：

```
printf("%d\n",(y*y+2*y)*2);
```

相当于计算：$2(y^2+2y)$。

但要注意的是，在宏定义中表达式(y*y+2*y)两边的括号不能少。否则会发生错误。如当作以下定义后：

```
#define M y*y+2*y
```

在宏展开时将得到下述语句：

```
printf("%d\n",y*y+2*y*2);
```

这相当于：y^2+4y。

显然与原题意要求不符。计算结果当然是错误的。因此在作宏定义时必须十分注意。应保证在宏代换之后不发生错误。

对于宏定义还要说明以下几点：

（1）宏定义是用宏名来表示一个字符串，在宏展开时又以该字符串取代宏名，这只是一种简单的代换，字符串中可以含任何字符，可以是常数，也可以是表达式，预处理程序对它不作任何检查。如有错误，只能在编译已被宏展开后的源程序时发现。

（2）宏定义不是说明或语句，在行末不必加分号，如加上分号则连分号也一起置换。

（3）宏定义必须写在函数之外，其作用域为宏定义命令起到源程序结束。如要终止其作用域可使用#undef 命令。

例如：

```
#define M (y*y+2*y)
main()
{
   …
}
#undef M
f1()
{
   …
}
```

表示 M 只在 main 函数中有效，在 f1 中无效。

（4）宏名在源程序中若用引号括起来，则预处理程序不对其作宏代换。

【例 1-4】无参宏定义的不正确使用示例程序。（ex1_4.cpp）

```
#define M (y*y+2*y)
main()
{
   printf("M");
   printf("\n");
}
```

上例中定义宏名 M，但在 printf 语句中 M 被引号括起来，表示把“M”当字符串处理，不作宏代换。因此程序的运行结果为 M。

（5）宏定义允许嵌套，在宏定义的字符串中可以使用已经定义的宏名。在宏展开时由预处理程序层层代换。

例如：

```
#define PI 3.1415926
#define S PI*r*r                // PI 是已定义的宏名
```

对语句：

```
printf("%f",S);
```

在宏代换后变为：

```
printf("%f",3.1415926*y*y);
```

（6）习惯上宏名用大写字母表示，以便于与变量区别。但也允许用小写字母。

（7）可用宏定义表示数据类型，以便书写方便。

例如：

```
#define STU struct stu
```

在程序中可用 STU 作变量说明：

```
STU body[5],*p;
#define INTEGER int
```

在程序中即可用 INTEGER 作整型变量说明：

```
INTEGER a,b;
```

应注意用宏定义表示数据类型和用 typedef 定义数据说明符的区别。

宏定义只是简单的字符串代换，是在预处理时完成的，而 typedef 是在编译时处理的，它不是作简单的代换，而是对类型说明符重新命名。被命名的标识符具有类型定义说明的功能。

请看下面的例子：

```
#define PIN1 int *
typedef (int *) PIN2;
```

从形式上看这两者相似， 但在实际使用中却不相同。下面用 PIN1、PIN2 说明变量时就可以看出它们的区别：

PIN1 a,b;在宏代换后变成:

```
int *a,b;
```

表示 a 是指向整型的指针变量，而 b 是整型变量。

然而：

```
PIN2 a,b;
```

表示 a 和 b 都是指向整型的指针变量。因为 PIN2 是一个类型说明符。由这个例子可见，宏定义虽然也可表示数据类型， 但毕竟是作字符代换。在使用时要分外小心，以避免出错。

（8）对输出格式作宏定义，可以减少书写麻烦。

【例 1-5】对输出格式作宏定义。（ex1_5.cpp）

```
#define P printf
#define D "%d\n"
#define F "%f\n"
main()
{
    int a=5, c=8, e=11;
    float b=3.8, d=9.7, f=21.08;
    P(D F,a,b);
    P(D F,c,d);
    P(D F,e,f);
}
```

2. 带参宏定义

C 语言允许宏带有参数。在宏定义中的参数称为形式参数，在宏调用中的参数称为实际参数。对带参数的宏，在调用中，不仅要宏展开，而且要用实参去代换形参。

带参宏定义的一般形式为：

```
#define   宏名(形参表)   字符串
```

在字符串中含有各个形参。

带参宏调用的一般形式为：

```
宏名(实参表);
```

例如：

```
#define M(y) y*y+2*y        //宏定义
…
k=M(5);                     //宏调用
…
```

在宏调用时，用实参 5 去代替形参 y，经预处理宏展开后的语句为：

```
k=5*5+2*5
```

【例 1-6】带参宏定义示例程序。（ex1_6.cpp）

```
#define MAX(a,b) (a>b)?a:b
main()
{
   int x,y,max;
   printf("请输入两个整数：");
```

```
    scanf("%d%d",&x,&y);
    max=MAX(x,y);
    printf("较大数为：%d\n",max);
}
```

上例程序的第一行进行带参宏定义，用宏名 MAX 表示条件表达式(a>b)?a:b，形参 a 和 b 均出现在条件表达式中。程序中 max=MAX(x,y)为宏调用，实参 x 和 y 将代换形参 a 和 b。宏展开后该语句为：

```
max=(x>y)?x:y;
```

用于计算 x、y 中的大数。

对于带参宏定义有以下问题需要说明：

（1）带参宏定义中，宏名和形参表之间不能有空格出现。

例如把

```
#define MAX(a,b) (a>b)?a:b
```

写为

```
#define MAX   (a,b)   (a>b)?a:b
```

将被认为是无参宏定义，宏名 MAX 代表字符串 (a,b) (a>b)?a:b。宏展开时，宏调用语句

```
max=MAX(x,y);
```

将变为

```
max=(a,b)(a>b)?a:b(x,y);
```

这显然是错误的。

（2）在带参宏定义中，形式参数不分配内存单元，因此不必作类型定义。而宏调用中的实参有具体的值。要用它们去代换形参，因此必须作类型说明。这是与函数中的情况不同的。在函数中，形参和实参是两个不同的量，各有自己的作用域，调用时要把实参值赋予形参，进行“值传递”。而在带参宏中，只是符号代换，不存在值传递的问题。

（3）在宏定义中的形参是标识符，而宏调用中的实参可以是表达式。

【例 1-7】带参宏定义示例程序。（ex1_7.cpp）

```
#define S(x) (x)*(x)
main()
{
   int a,b;
   printf("请输入一个整数：");
   scanf("%d",&a);
   b=S(a+1);
   printf("b=%d\n",b);
}
```

例 1-7 中第一行为宏定义，形参为 x。程序 b=S(a+1)宏调用中实参为 a+1，是一个表达式，在宏展开时，用 a+1 代换 x，再用(x)*(x) 代换 S，得到如下语句：

```
s=(a+1)*(a+1);
```

这与函数的调用是不同的，函数调用时要把实参表达式的值求出来再赋予形参。而宏代换中对实参表达式不作计算直接地照原样代换。

（4）在宏定义中，字符串内的形参通常要用括号括起来以避免出错。在上例中的宏定义中(x)*(x)表达式的 x 都用括号括起来，因此结果是正确的。如果去掉括号，把宏定义改为:

```
#define S(x) x*x
```

宏代换后将得到以下语句：

```
s=a+1*a+1;
```

这显然与题意相违，因此参数两边的括号是不能少的。

（5）带参的宏和带参函数很相似，但有本质上的不同。

（6）宏定义也可用来定义多个语句，在宏调用时，把这些语句又代换到源程序内。看下面的例子。

【例 1-8】多语句宏定义示例程序。（ex1_8.cpp）

```
#define SSSV(s1,s2,s3,v) s1=l*w;s2=l*h;s3=w*h;v=w*l*h;
main()
{
   int l=3,w=4,h=5,sa,sb,sc,vv;
   SSSV(sa,sb,sc,vv);
   printf("sa=%d\nsb=%d\nsc=%d\nvv=%d\n",sa,sb,sc,vv);
}
```

程序第一行为宏定义，用宏名 SSSV 表示 4 个赋值语句，4 个形参分别为 4 个赋值符左部的变量。在宏调用时，把 4 个语句展开并用实参代替形参。使计算结果送入实参之中。

1.3.2 文件包含

文件包含是 C 预处理程序的另一个重要功能。

文件包含命令行的一般形式为：

```
#include "文件名"
```

在前面已多次用此命令包含过库函数的头文件。例如：

```
#include "stdio.h"
#include "math.h"
```

文件包含命令的功能是把指定的文件插入该命令行位置取代该命令行，从而把指定的文件和当前的源程序文件连成一个源文件。

在程序设计中，文件包含是很有用的。一个大的程序可以分为多个模块，由多个程序员分别编程。有些公用的符号常量或宏定义等可单独组成一个文件，在其他文件的开头用包含命令包含该文件即可使用。这样，可避免在每个文件开头都去书写那些公用量，从而节省时间，并减少出错。

对文件包含命令还要说明以下几点：

（1）包含命令中的文件名可以用双引号括起来，也可以用尖括号括起来。例如以下写法都是允许的：

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

但是这两种形式是有区别的：使用尖括号表示在包含文件目录中去查找（包含目录是由用户在设置环境时设置的），而不在源文件目录去查找。

使用双引号则表示首先在当前的源文件目录中查找，若未找到才到包含目录中去查找。用户编程时可根据自己文件所在的目录来选择某一种命令形式。

（2）一个 include 命令只能指定一个被包含文件，若有多个文件要包含，则需用多个 include 命令。

（3）文件包含允许嵌套，即在一个被包含的文件中又可以包含另一个文件。

1.3.3 条件编译

预处理程序提供了条件编译的功能。可以按不同的条件去编译不同的程序部分，因而产

生不同的目标代码文件。这对于程序的移植和调试是很有用的。

条件编译有三种形式，下面分别进行介绍。

第一种形式：

```
#ifdef  标识符
    程序段 1
#else
    程序段 2
#endif
```

它的功能是，如果标识符已被 #define 命令定义过则对程序段 1 进行编译；否则对程序段 2 进行编译。如果没有程序段 2（它为空），本格式中的#else 可以没有，即可以写为

```
#ifdef  标识符
    程序段
#endif
```

【例 1-9】条件编译#ifdef 命令的使用示例程序。（ex1_9.cpp）

```
#define DEBUG                   //无参宏定义
main()
{
  #ifdef DEBUG                  //条件编译
        printf("Debugging...\n");
  #else
        printf("Running...\n");
  #endif
}
```

由于在程序中插入了条件编译预处理命令，因此要根据 DEBUG 是否被定义过来决定编译哪一个 printf 语句。而在程序的第一行已对 DEBUG 作过宏定义，因此应对第一个 printf 语句作编译，跳过第二个 printf 语句，故运行结果是：Debugging...。

第二种形式：

```
#ifndef  标识符
    程序段 1
#else
    程序段 2
#endif
```

与第一种形式的区别是将 ifdef 改为 ifndef。它的功能是，如果标识符未被#define 命令定义过则对程序段 1 进行编译，否则对程序段 2 进行编译。这与第一种形式的功能正相反。

第三种形式：

```
#if  常量表达式
    程序段 1
#else
    程序段 2
#endif
```

它的功能是，如常量表达式的值为真（非 0），则对程序段 1 进行编译，否则对程序段 2 进行编译。因此可以使程序在不同条件下完成不同的功能。

【例 1-10】根据宏定义选择条件编译计算圆面积或矩形面积。（ex1_10.cpp）

```
#define PI 3.1415
#define R 1
main()
```

```
{
    float c,r,s;
    printf ("input a number:    ");
    scanf("%f",&c);
    #if R
        r=PI*c*c;
        printf("圆面积是: %f\n",r);
    #else
        s=c*c;
        printf("矩形面积是: %f\n",s);
    #endif
}
```

程序运行结果如图 1-4 所示。

```
input a number:  3.2
圆面积是: 32.168961
```

图 1-4　例 1-10 程序的运行结果

本例中采用了第三种形式的条件编译。在程序第一行宏定义中，定义 R 为 1，因此在条件编译时，常量表达式的值为真，故计算并输出圆面积。

上面介绍的条件编译当然也可以用条件语句来实现。 但是用条件语句将会对整个源程序进行编译，生成的目标代码程序很长，而采用条件编译，则根据条件只编译其中的程序段 1 或程序段 2，生成的目标程序较短。如果条件选择的程序段很长，采用条件编译的方法是十分必要的。

1.4　程序设计基础

1.4.1　算法概念

算法是指解决问题的方法和步骤。编写程序是让计算机解决实际问题，是算法的程序实现。一般编制正确的计算机程序必须具备两个基本条件：一是掌握一门计算机高级语言的规则，二是要掌握解题的方法和步骤。

计算机语言只是一种工具。简单地掌握语言的语法规则是不够的，最重要的是学会针对各种类型的问题拟定出有效的解题方法和步骤的算法。

正确的算法具有以下 5 个特性：

（1）有穷性。一个算法必须总是在执行有穷步之后结束，且每一步都在有穷时间内完成。

（2）确定性。算法中每一条指令必须有确切的含义。不存在二义性，且算法只有一个入口和一个出口。

（3）可行性。一个算法是可行的，即算法描述的操作都是可以通过已经实现的基本运算执行有限次来实现的。

（4）输入。一个算法有零个或多个输入，这些输入取自于某个特定的对象集合。

（5）输出。一个算法有一个或多个输出，这些输出是同输入有着某些特定关系的量。

下列过程就不是一个正确的算法：

第 1 步：令 n 等于 0。

第 2 步：n 加 1。

第 3 步：转向第 2 步。

如果利用计算机执行此过程，从理论上讲，计算机将永远执行下去，即死循环。

而下列过程就是一个正确的算法：

第 1 步：令 n 等于 0。

第 2 步：n 加 1。

第 3 步：如果 n 小于 100，则转向第 2 步；否则停止。

实质上，算法反映的是解决问题的思路。许多问题，只要仔细分析对象数据，就容易找到处理方法。

1.4.2　算法的表示

算法的表示方法有很多，主要有传统流程图、N-S 图、伪代码、自然语言和计算机程序语言等。这里重点介绍传统流程图和 N-S 图。

1. 传统流程图

用图形表示算法，直观形象，易于理解。流程图是用一些图框来表示各种操作。美国国家标准化协会 ANSI 规定了一些常用的流程图符号，如图 1-5 所示。

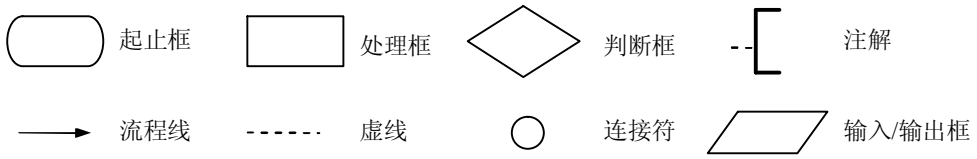


图 1-5　流程图符号

图 1-5 中菱形框的作用是对一个给定的条件进行判断，根据给定的条件是否成立来决定如何执行其后的操作。它有一个入口、两个出口，其流程如图 1-6 所示。

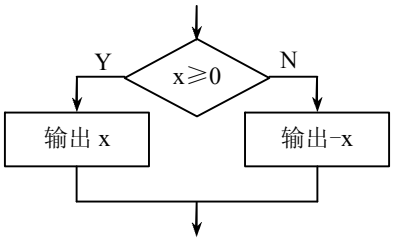


图 1-6　条件判断示意图

菱形框两侧的 Y 和 N 表示“是”（YES）和“否”（NO）。

【例 1-11】画出求 1+2+3+…+100 之和的流程图。流程图如图 1-7 所示。

2. N-S 图

1973 年美国学者提出了一种新的流程图形式，即 N-S 图。在这种流程图里，完全去掉了带箭头的流程线。全部算法写在一个矩形框内，在框内还可以包含其他从属于它的方框，即由一些基本的框组成一个大框。这种流程图适于结构化程序设计算法的描述。

N-S 图有以下流程图符号：

（1）顺序结构。用图 1-8 形式表示。A 和 B 两个框表示了顺序结构。

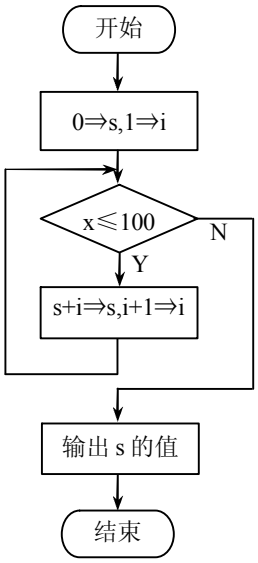


图 1-7　例 1-11 的流程图

（2）选择结构。用图 1-9 形式表示。当 P 条件成立时执行 A 操作，当 P 条件不成立时执行 B 操作。

（3）循环结构。循环结构分为当型循环和直到型循环，当型循环如图 1-10 所示，当条件 P 成立时反复执行 A 操作，当条件 P 不成立时结束循环；直到型循环结构如图 1-11 所示，反复执行 A 操作，直到条件 P 成立。实际上也是当 P 不成立时退出循环，只是 A 至少执行一次。

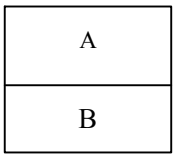


图 1-8　顺序结构

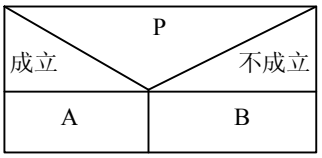


图 1-9　选择结构

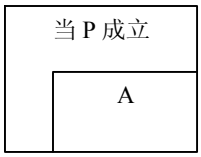


图 1-10　当型循环

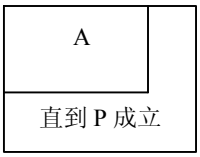


图 1-11　直到型循环

例 1-11 的 N-S 图如图 1-12 所示。

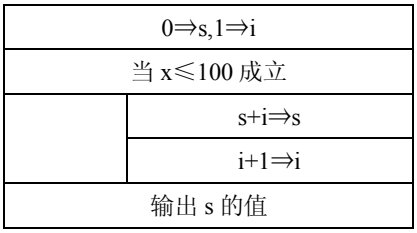


图 1-12　N-S 图

3．文字描述法

用自然语言描述法，用自然语言表示的算法通俗易懂，但易引起“歧义性”。除了很简单的问题，一般不用自然语言表示算法。

例如：有 50 个学生，要求将他们之中成绩在 80 分以上者打印出来。用 g 代表学生成绩，g_i 代表第 i 个学生成绩，算法可表示如下：

S1：1←i

S2：如果 g_i >=80，则打印 g_i，否则不打印

S3：i←i+1

S4: 如果 i<=50，返回 S2，继续执行； 否则，算法结束。

用计算机语言表示算法必须严格遵循所用语言的语法规则。

4．伪代码表示法

伪代码使用介于自然语言和计算机语言之间的文字和符号来描述算法。它的控制结构往往类似于 Pascal、Borland C++、Visual C++等程序语言，但其中可使用任何表达能力强的方法使算法表达更加清晰和简洁，而不至于陷入具体的程序语言的某些细节。

1.4.3　程序的三种基本结构

从程序流程的角度来看，程序可以分为三种基本结构， 即顺序结构、选择结构、循环结构。这三种基本结构可以组成各种复杂程序。C 语言提供了多种语句来实现这些程序结构。

1．顺序结构

顺序结构是最简单的基本结构，要求顺序地执行且必须执行由先后顺序排列的每一个最基本的处理单位。

顺序结构的示意图如图 1-13 所示，表示先执行“语句 1”，然后再顺序执行“语句 2”。

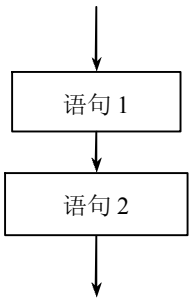


图 1-13　顺序结构

2．选择结构

选择结构又称作分支结构。在选择结构中，要根据逻辑条件的成立与否，分别选择执行不同的语句。

选择结构的示意图如图 1-14 所示，表示当逻辑条件成立时，执行语句 A，否则执行语句 B。

3．循环结构

循环结构的执行方式是根据某项条件重复地执行一些语句若干次直到某条件成立或不成立为止。循环结构分为以下两种形式：

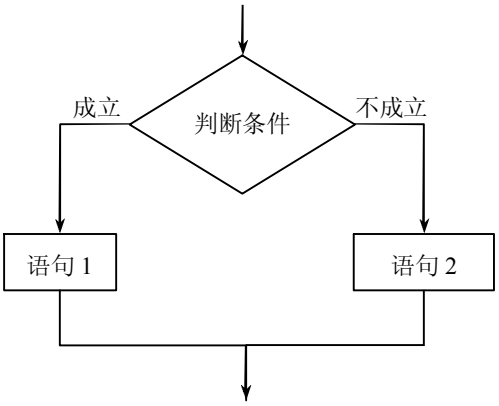
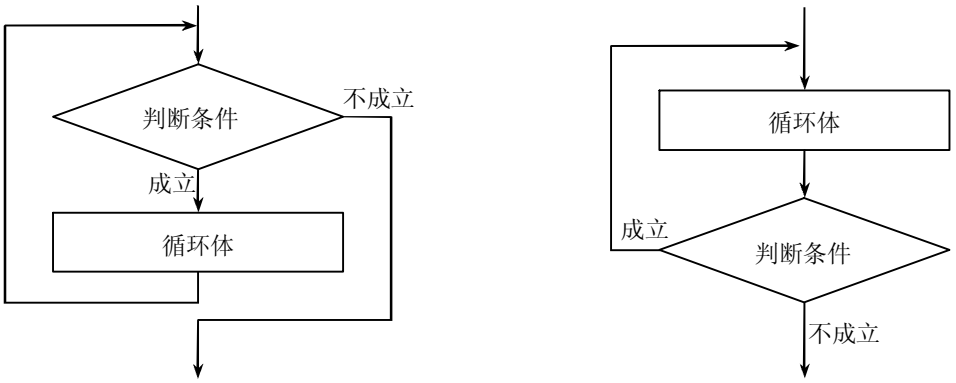


图 1-14　选择结构

（1）当型循环。在如图 1-15（a）所示的当型循环结构中，当逻辑条件成立时，就反复执行循环体，直到逻辑条件不成立时结束。

（2）直到型循环。在如图 1-15（b）所示的直到型循环结构中，反复执行循环体，直到逻辑条件不成立时结束。



（a）当型循环结构　　（b）直到型循环结构

图 1-15　循环结构

三种基本结构的共同特点：

- 只有一个入口；
- 只有一个出口；
- 结构内的每一部分都有机会被执行到；
- 结构内不存在“死循环”。

1.4.4　程序的书写风格和书写格式

从书写清晰，便于阅读、理解、维护的角度出发，在书写程序时应遵循以下规则：

（1）一个说明或一个语句占一行。

（2）用{} 括起来的部分，通常表示了程序的某一层次结构。{}一般与该结构语句的第一个字母对齐，并单独占一行。

（3）低一层次的语句或说明可比高一层次的语句或说明缩进若干格后书写。以便看起来

更加清晰，增加程序的可读性。

在编程时应力求遵循这些规则，以养成良好的编程风格。

程序的结构仅由三种基本结构组合、嵌套而成，且满足：

（1）每个程序模块只有一个入口和一个出口；

（2）没有死语句（永远执行不到的语句）；

（3）没有死循环（永远执行不完的无终止的循环）；

（4）程序的书写必须按一定的规范和格式进行，不能随心所欲地拼凑。

1.5　Visual C++ 6.0 编程环境

在写好一个 C 程序后，要经过以下步骤才能在计算机上运行，如图 1-16 所示。

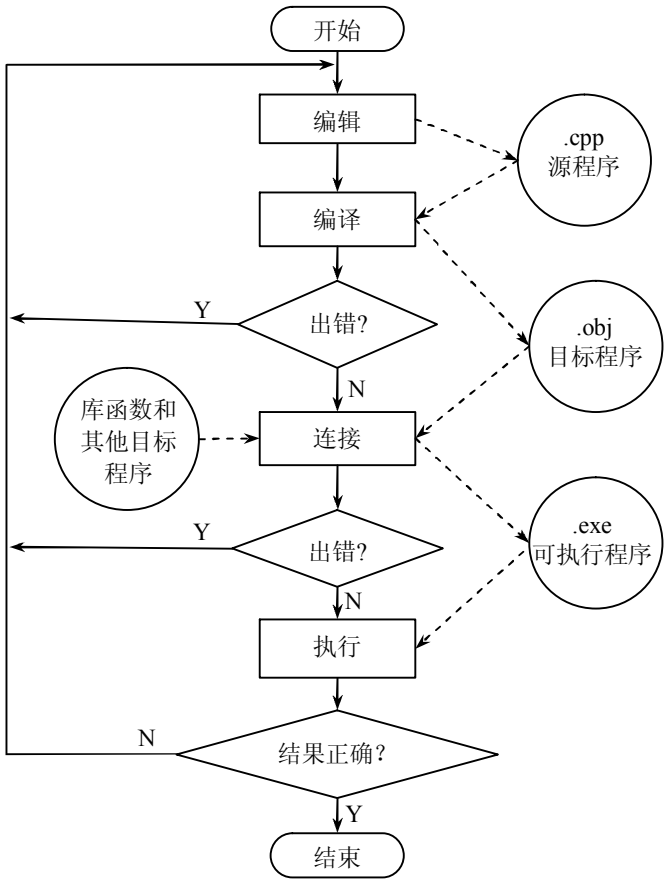


图 1-16　C 程序开发步骤

（1）编辑：选择适当的编辑程序（如 Visual C++ 6.0），将 C 语言源程序通过键盘输入到计算机中，并以文件的形式存入到磁盘中（.cpp）。

（2）编译：将源程序翻译成二进制形式的目标程序（.obj）。

（3）连接：编译后生成的目标文件与系统的函数库和其他目标程序，经过连接后生成最终的可执行程序（.exe）。

Visual C++ 6.0 是美国微软公司开发的 C++集成开发环境，它集源程序的编写、编译、连接、调试、运行，以及应用程序的文件管理于一体，是当前 PC 上最流行的 C 程序开发环境。Visual C++ 6.0 也可以编写控制台程序，系统中也包含 C 语言的编译器，可以用来编译 C 程序，不过要求源程序文件的扩展名必须是.cpp。

1. Visual C++ 6.0 界面

Visual C++ 6.0 集成开发环境被划分成 4 个主要区域：菜单栏和工具栏、项目工作区、代码编辑窗格和输出窗格，如图 1-17 所示。

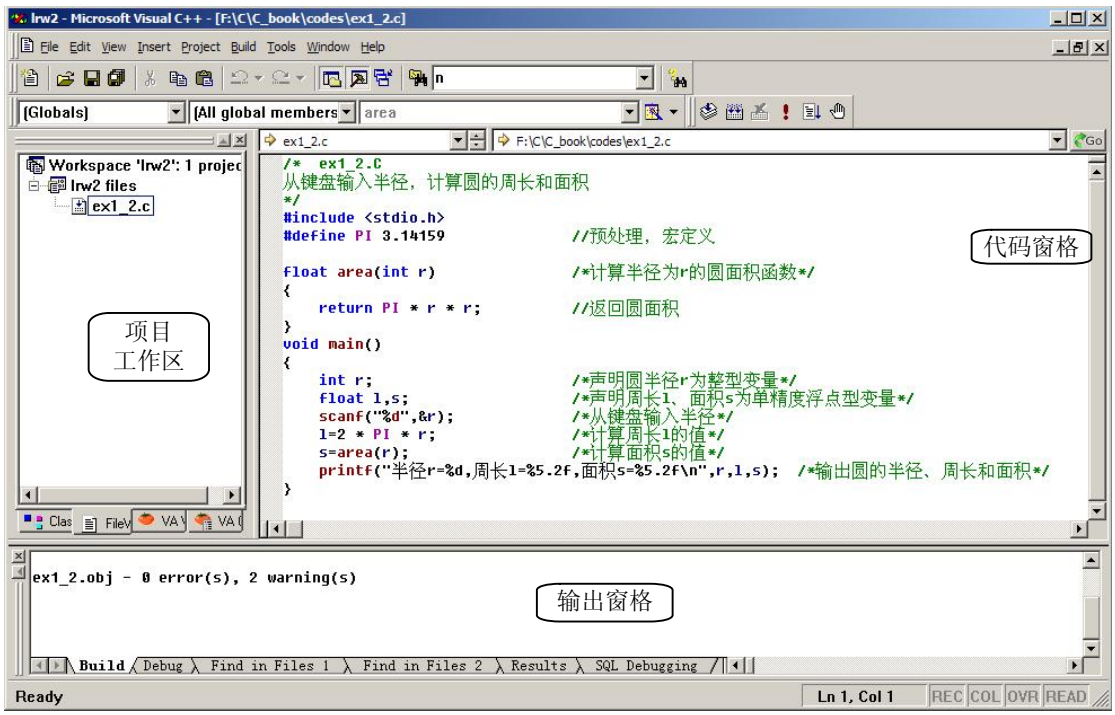


图 1-17　Visual C++集成开发环境

（1）菜单栏。Visual C++菜单栏包含了开发环境中几乎所有的命令，它为用户提供了代码操作，程序的编译、调试，窗口操作等一系列的功能。与一般 Windows 应用程序一样有 File、Edit、View、Insert、Project、Build、Tools、Window、Help 等菜单。

（2）工具栏。通过工具栏，可以迅速地使用常用的菜单命令。最常用的工具栏是标准工具栏，当鼠标指向这些工具时，通常有提示工具含义的信息，因此也比较容易掌握。若要显示或隐藏某个工具栏，则在任一工具栏的快捷菜单中选择相应的命令即可。

（3）项目工作区。项目是开发一个程序时需要的所有文件的集合，而工作区是进行项目组织的工作空间。利用项目工作区可以观察和存取项目的各个组成部分。在 Visual C++中，一个工作区可以包含多个项目。

项目工作区有 Class View 和 File View 选项卡，Class View 显示当前项目的类，全局的变量和函数也在这里显示。File View 显示当前项目的源文件、头文件、资源文件等。

在 Visual C++中，项目中所有的源文件都是采用文件夹的方式进行管理的，它将项目名作为文件夹，在此文件夹下包含源程序代码文件（.cpp、.h）、项目文件（.dsp）以及项目工作区文件（.dsw）等。若要打开一个项目，只需打开对应的项目工作区文件即可。

（4）代码窗格。一般位于开发环境中的右边，各种程序代码的源文件、资源文件、文档文件等都可以通过该窗格显示。

（5）输出窗格。输出窗格有多个选项卡，最常用的是“编译”。在编译、连接时，这里会显示有关的信息，供调试程序用。

（6）状态栏。状态栏一般位于开发环境的最底部，它用来显示当前操作状态、注释、文本光标所在的行和列号等信息。

2. C 程序的开发过程

在 Visual C++中，一个简单 C 程序的编写、运行过程是：创建一个空工程→创建一个 C 源文件，输入源程序→进行编译、连接、运行。

操作步骤如下：

1）创建空工程

（1）选择 File→New 命令。

（2）选择 Projects 选项卡，选择 Win32 Console Application（32 位控制台应用程序），输入工程名 p1_2，确保单选按钮 Create new workspace 被选定，输入工程位置 F:\C\C_BOOK\p1_2，注意 F:\C\C_BOOK 文件夹需要事先建好，如图 1-18 所示。

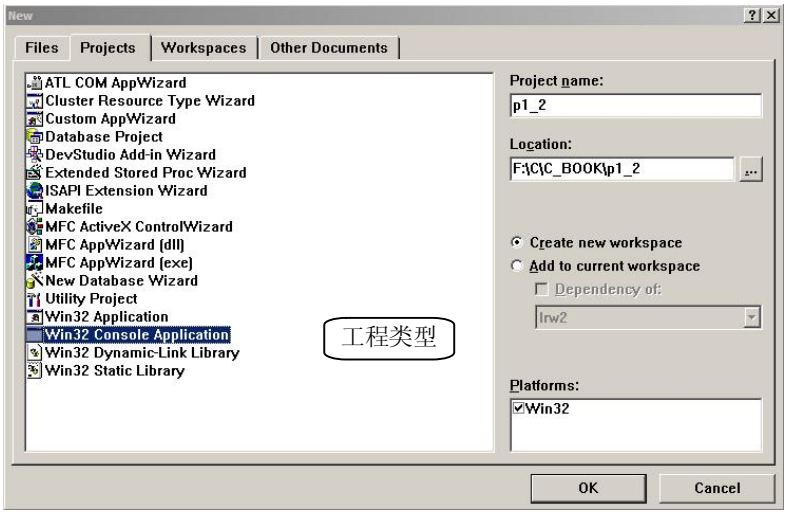


图 1-18　New 对话框

（3）在随后弹出的向导对话框中，选择 An empty project，并单击 Finish 按钮，显示新建工程的有关信息。

（4）单击 OK 按钮，创建空工程的工作结束。

此时为工程 p1_2 创建了 F:\C\C_BOOK\p1_2 文件夹，并在其中生成了文件 p1_2.dsp、p1_2.dsw 和文件夹 Debug，如图 1-19 所示。Debug 文件夹用于存放编译、连接过程中产生的文件。

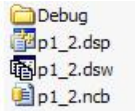


图 1-19　生成文件夹和文件

2）创建 C 源文件

（1）选择 File→New 命令。

（2）选定 Files 选项卡，选定 C++ Source File，并输入源程序文件名 ex1_2.c，如图 1-20 所示。

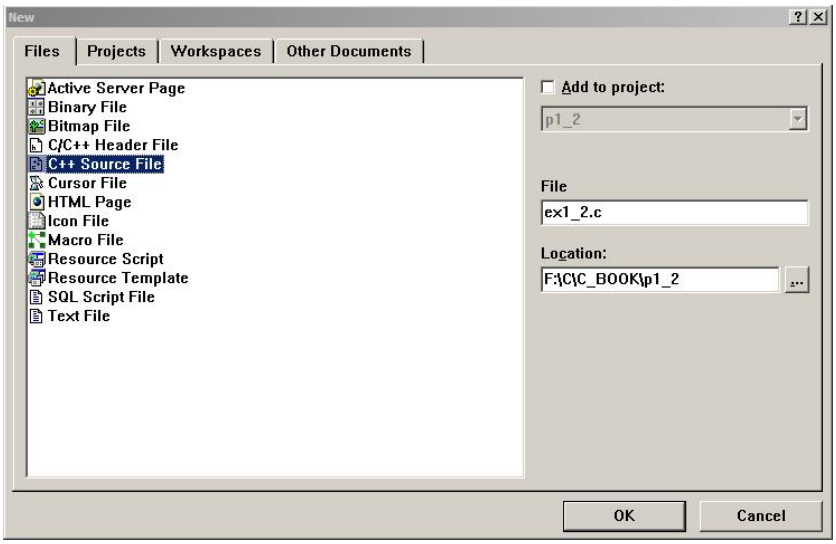


图 1-20　新建 C++ Source File

（3）输入、编辑源程序。

在这个阶段中，F:\C\C_BOOK\p1_2 文件夹中创建了 ex1_2.cpp。

（4）编译、连接和运行。

选择 Build→Execute p1_2.exe 命令进行编译、连接和运行，会在输出窗格中显示有关信息，如图 1-21 所示。若程序有错，则继续编辑。

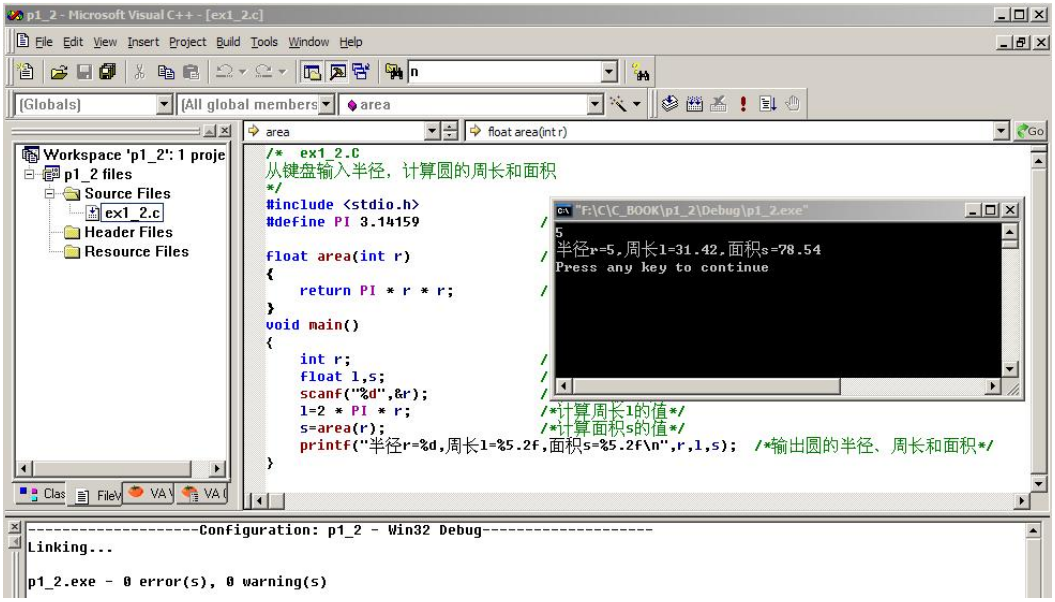


图 1-21　编译和运行的界面

编译、连接和运行可以分别执行。

（1）编译（Ctrl+F7）。选择 Build→Compile ex1_2.cpp 命令，编译结果显示在输出窗格中，如果没有错误，则生成 ex1_2.obj。

（2）连接（F7）。选择 Build→Build p1_2.exe 命令，连接信息显示在输出窗格中，如果没有错误，则生成 p1_2.exe。

（3）运行（Ctrl+F5）。选择 Build→Execute p1_2.exe 命令。

e1_1 工程的文件夹如图 1-22 所示。

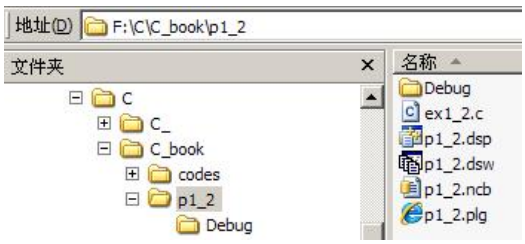


图 1-22　e1_1 工程的文件夹

在 F:\C\C_book\p1_2\Debug 中生成了 p1_2.obj、p1_2.exe 等文件，如图 1-23 所示。p1_2.obj 是编译后产生的目标代码文件，p1_2.exe 是最终生成的可执行文件。

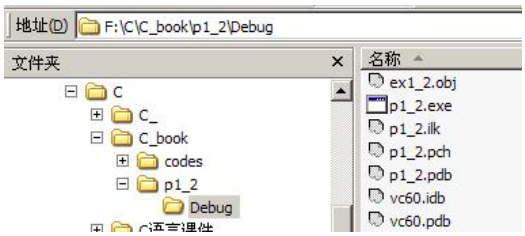


图 1-23　p1_2 工程调试后的 Debug 文件夹

至此，一个简单 C 程序的编写、调试过程结束。

在此例中，ex1_2.cpp 文件是最重要的一个文件，源程序保存在这个文件中，其他文件一般都是系统自动生成的。但是，在 Visual C++中，仅有.cpp 文件是不能直接编译、连接的，需要首先用“构建”命令让系统自动创建一个工程并将 ex1_2.cpp 文件加入到该工程中，然后才能执行各种操作。因此，程序员可以只复制.cpp 文件，若要复制整个工程的文件夹，推荐通过执行 Build（组建）菜单中的 Clear（清除）命令以删除 Debug 文件夹中在编译过程中生成的中间文件及二进制文件，因为它们占据相当大的存储空间。

1.6　本章小结

C 语言自 1972 年投入使用以来，是当今使用最为广泛的程序设计语言之一。C 语言具有简洁、灵活，运算符和数据类型丰富等特点；一个正确的 C 语言程序由一个主函数和若干个子函数组成，从主函数开始运行，最后在主函数中结束。

（1）预处理功能是 C 语言特有的功能，它是在对源程序正式编译前由预处理程序完成的。程序员在程序中用预处理命令来调用这些功能。

（2）宏定义是用一个标识符来表示一个字符串，这个字符串可以是常量、变量或表达式。在宏调用中将用该字符串代换宏名。

（3）宏定义可以带有参数，宏调用时是以实参代换形参。而不是“值传送”。

（4）为了避免宏代换时发生错误，宏定义中的字符串应加括号，字符串中出现的形式参数两边也应加括号。

（5）文件包含是预处理的一个重要功能，它可用来把多个源文件连接成一个源文件进行编译，结果将生成一个目标文件。

（6）条件编译允许只编译源程序中满足条件的程序段，使生成的目标程序较短，从而减少了内存的开销并提高了程序的效率。

（7）使用预处理功能便于程序的修改、阅读、移植和调试，也便于实现模块化程序设计。

算法是指解决问题的方法和步骤，是程序设计的精华和核心，一个算法具有有穷性、确定性、输入/输出和可行性等特征；算法描述工具很多，主要有传统流程图、N-S 图、伪代码和计算机程序语言等。

本章还介绍了 C 语言的编程环境 Visual C++ 6.0，作为 C 语言的学习，最好熟练使用这个环境。

习题一

1．一个 C 程序是从_____函数开始执行的。

2．一个 C 源程序中的基本单位是_____。

3．一个 C 语言程序是由_____组成。

4．结构化程序设计中的三种基本结构是_____、_____、_____。

5．C 语言的主要特点是什么？

6．下面程序的运行结果是_____。

```
#define DOUBLE(r) r*r
main()
{   int x=1 ,y=2,t;
    t=DOUBLE(x+y);
    printf("%d\n",t);
}
```

7．下面程序的运行结果是_____。

```
#define MUL(z) (z)*(z)
main()
{
      printf("%d\n",MUL(1+2)+3);
}
```

8．下面程序的运行结果是_____。

```
#define SELECT(a,b)a<b?a:b
main()
{ int m=2, n= 4;
printf("%d\n",SELECT(m,n));
}
```

9．请参照本章例题编写一个 C 程序，输出以下信息：

```
******************************
    This is my first c program.
******************************
```

10．编写一个程序，从键盘输入三个整数 a、b、c，计算表达式 a+b*c 的值，并输出计算结果。

11．C 语言源程序文件、目标文件和可执行文件的扩展名是什么？

12．用 Visual C++ 6.0 调试本章的例题程序。