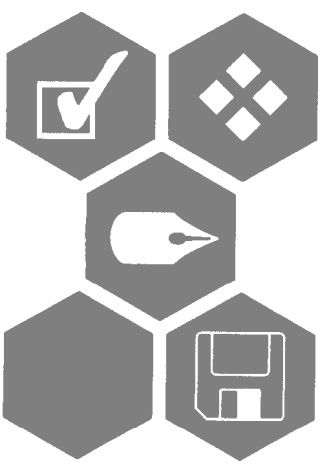
1

简单的 C 程序设计



学习新的程序设计语言的最佳途径是尽早地用它编写程序、进行程序的调试，解决实际问题。本章从最简单的第一个程序开始，逐步介绍了 C 语言基于函数的程序结构，变量与常量、算术运算、循环结构、基本输入输出标准函数，使读者了解一个 C 语言程序的基本框架和它的书写格式。至于有关语法规则细节，读者先不必深究，学到有关章节时自然会理解。通过介绍 Turbo C 集成环境的使用，要求读者掌握一个 C 程序的编写、编译、连接、调试、直到成功运行的全过程，并能够动手上机操作。由于所举例子并没有用到 C语言的所有特性，不可能完整地表达出使用 C 语言编程的特点，所以本章最后简要阐述了 C 语言的主要特点，为读者学习以下各个章节起一个点睛的作用，相信读者通过后面各章节的学习，能够真正理解到这些特点。

1.1 几个简单的 C 程序

例 1.1：在 DOS 屏幕上显示“ hello,world！ ”

```
/*                第一个 C 语言程序举例                */
/*    包含有关标准库的信息    */
#include <stdio.h>
/*    定义名为 main 的函数，它不接收实参值*/
main()
{
/*    main 的语句括在花括号中    */
/*    main 函数调用库函数 printf 原样打印字符序列，\n 代表换行    */
printf( "word, hello! \n" );
}
```

程序分析：

1. 注解：夹在“ /* ”与“ */ ”之间的字符序列，用于解释该程序是做什么的，目的是为了使程序更易于理解和提起记忆，这些注解在编译时会被自动忽略掉。它们可以在程序中自由地使用，可以出现在程序中的任何位置。字符序列中可以使用空格、制表符或换行符。读者应重视使用注解，养成良好的编程习惯。
2. main 函数：函数是 C 语言的基本单位，每一个 C 程序，不论大小，都是由一个或多个函数组成的。函数是一个单独的程序模块，完成指定的功能。在本例中具有两个函数，一个是名字为 main 的函数。一般而言，可以给函数任意命名，但 main 是一个特殊的函数名。一个 C 程序不论由多少个文件组成，都有一个且只能有一个 main 函数，通常称为主函数。任何一个 C 程序都从它开始运行。main 函数常常还要调用其他函数来协助其完成某些工作，被调用的函数有些是由程序人员自己编写的,有些则由系统函数库提供。本例中的 printf 函数就是由系统函数库提供的。函数中的语句用一对花括号 { }括起来。本例中的 main 函数只包含一条语句 printf("world，hello！\n");，语句最后有一个分号。
3. 函数的调用：当调用一个函数时，先要给出这个函数的名字，然后考虑与被调函数的数据交换。在函数之间进行数据交换的一种方法是让调用函数向被调用函数提供

一串叫做实参（变元）的值。函数名后面的一对圆括号用于把这一串实参（实参表）括起来。在本例子中，main 函数不要求任何实参，故用空实参表（）表示。main 函数用“ world,hello!\n ”作实参调用 printf 函数。

4. printf 是一个用于打印的格式化输出库函数，在本例中，它用于在 DOS 屏幕上照原样显示双引号内的字符序列“ world，hello！\n” （不包括双引号）。用双引号括住的字符序列叫做字符串。本例中仅使用字符串作为 printf 的实参。

5. 字符串中的字符序列 \n 表示换行符，在显示时它用于指示从下一行的左边开始显示字符。

注意，\n 只表示一个字符。除此之外，C 语言中还有：表示制表符的\t 、表示回退符的\b、表示双引号的 \"、表示反斜杠符本身的 \\。

6. 文件包含命令：#include <stdio.h>。

这里的 #include 称为文件包含命令，其意义是把尖括号 < > 内指定的文件包含到本程序中，成为本程序的一部分。被包含的文件通常是由系统提供的，其扩展名为“.h”。因此也称为头文件或首部文件。如果使用了系统提供的库函数，一般应在文件的开始用 #include 命令，将被调用的库函数信息包含到本文件中。本例中的 #include <stdio.h> 是因为调用了标准输入输出库中的 printf 。需要说明的是，C 语言规定对 scanf 和 printf 这两个函数可以省去对其头文件的包含命令。所以在本例中也可以删去第二行的包含命令#include <stdio.h>。

例1.2：编写程序，计算 t 的值

$$t=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}$$

这个程序本身仍只由一个名为 main 的函数组成。它要比上面一个例子长，增加了一些新的内容，包括变量说明、算术表达式以及格式输出。该程序如下：

```
#include <stdio.h>
main ()
{
/*  定义整型变量          */
int sum;
   /*  计算累加和的语句     */
sum = 1 + 1/2 + 1/3 + 1/4 + 1/5;
   /*  按整型数输出计算结果   */
printf( "The sum is %d\n", sum)   ;
}
```

程序分析：

1. 变量说明：本例子的主函数体中分为两部分：一部分是说明部分，另一部分是执行部分。在说明部分说明了函数所用到的变量的类型，通常放在函数开始处的可执行语句之前。上一个例子没有使用任何变量，因此没有说明部分。 C 语言规定，程序中所有用到的变量都必须先说明，后使用，否则将会出错。说明语句由一个类型名与若干所要说明的变量名组成， int sum 中的 int 是类型名，sum 是变量名。

类型 int 表示所列变量为整型变量（整数不包含小数部分）。 float型表示所列变量为浮点变量（浮点数可以有小数部分）。除 int 与 float 之外，C 语言还提供了其他一些基本数据类型，包括：char （字符型）、short（短整型）、long （长整型）、double （双精度浮点型）。另外，还有由这些基本类型构成的数组、结构与联合类型、指向这些类型的指针类型以及返回这些类型的函数，将在后面适当的章节分别介绍它们。

2. 赋值表达式：与 Basic、Pascal、Fortran 语言一样，数值的计算使用赋值表达式实现。在 C 中“ = ”称为赋值号，含义不同于数学中的等号，而是将其右边表达式的值赋给左边的变量。

3. 算术表达式：书写的数学计算式在 C 语言中要写成合法的 C 语言算术表达式。在 C 中，算术运算符包括 “+’, 、“ -，，、“ * ”、 “/’，以及取模运算符‘ % ”。“ /,,整数除法要截取掉结果中的小数部分。表达式“ x % y ”的结果是“ x” 除以“ y”的余数，要求“ x ”和“ y ”均为整数，余数的符号与被除数相同。顺便提醒的是，C 语言中的乘号要写成‘ * ”，“1.0/2.0 ”表示浮点数的 1 除以浮点数的 2，而“1/2”表示整型数 1 除以整型数 2，它们的结果是不一样的。进一步的内容在下一章中介绍。

4. 格式输出：这个例子使用了 printf 函数更多的功能。Printf 是一个通用格式化输出函数，下一章将做详细介绍。本例中 printf 函数具有两个实参，第一个实参是要打印的字符串，其中百分号“ % ”指示用第二个实参 sum 对其进行替换， “d ” 指示按整型数打印 sum 的值。

5. 本例运行以后，在 DOS 屏幕上显示如下：

```
The sum is 1
```

显然，结果是不精确的。为什么呢？原因在于 1/2、1/3、1/4 和 1/5 在 C 中计算结果都是 0。为了得到更加精确的计算结果，必须用浮点数代替上面的整型数。

例1.3：修改后的程序

```
#include <stdio.h>
main ()
{
/*  定义单精度浮点型变量  */
float sum;
 /*计算累加和语句 */
sum = 1.0 + 1.0/2.0 + 1.0/3.0 + 1.0/4.0 + 1.0/5.0;
 /*  按浮点型数输出计算结果  */
printf( "The sum is %f\n", sum);
 }
```

程序分析：

1. 上一例不精确是因为 1/2 按整数除法，截取后结果为 0 . 1.0/2.0 是两个浮点数的除法，不作截取处理。在 C 语言程序中，浮点数最好写成带小数点，即使该浮点数取的是整数值，因为这样使程序比较清晰。

2. %f 对应于单精度的浮点数 sum 。表示按单精度浮点数格式打印计算结果。其他还有几种指定打印宽度的格式串：

%8d 打印十进制整数，至少 8 个字符宽。

%8f 打印浮点数，至少 8 个字符宽。

% .2f 打印浮点数，小数点后有两位小数。

%8.2f 打印浮点数，至少 8 个字符宽，小数点后有 2 位小数。

此外，printf 函数还可以识别如下格式说明：表示八进制数的 %o 、表示十六进制数的 %x 、表示字符的 %c 、表示字符串的%s 以及表示百分号 % 本身的 %%.

printf 函数第 1 个实参中的各个 % 分别对应于第 2 个、第 3 个…第 *n* 个实参，它们在数目和类型上都必须匹配，否则将出现错误。

例1.4:计 算 t 的值的另一个 C 程序

对于一个特定任务，可以用多种方法来编写程序。上面的程序直接按照算式写法表达，语句很长，可以想象，如果累加 100项，累加 1000项就很难写了。下面是另一版本的 C程序，完成同样的计算任务。

```
#include <stdio.h>
main ()
 {
int i;
float sum ;
sum = 1.0;
for( i=2; i<6 ; i++ )
   sum = sum + 1.0/i;
printf( "The sum is %f\n", sum);
 }
```

程序分析：

1. for 循环语句：本例的特点是累加计算，所以可以采用循环语句来实现。for 是一种循环语句，for 后面的圆括号内共包含三个部分，它们之间用分号隔开。第一部分 i = 2 是初始化部分，仅在进入循环前执行一次。第二部分是用于控制循环的条件测试部分 i<6，这个条件要进行求值。如果所求得的值为真，那么就执行循环体（本例循环体中只包含一条语句 sum = sum + 1.0/i;）,然后再执行第三部分 i++(i=i+1)，加步长，并再次对条件求值。一旦求得的条件值为假，那么就终止循环的执行。 for 循环语句的循环体可以是单条语句，也可以是用花括号括住的一组语句。
2. 条件测试：用于控制循环执行次数的条件测试 i<6 在 C 中称作条件表达式。如果 i 小于 6，其结果是“ 真 ”（True），否则是“ 假 ”（False），除小于 < 外，还有大于 > 、小于等于 <= 、大于等于 >= 、等 于 = 和不等于!= 等运算符，例如 i 不等于 10 写作 i != 10。

关于循环结构的用法将在第 3 章介绍。

3. 如果某个算术运算符的运算分量都是整数类型，那么就执行整数运算。如果某个算术运算符的运算分量有一个是浮点运算分量和一个是整数类型分量，那么这个整数类型分量在开始运算之前会被转换成浮点类型。所以 1.0/i 不会影响结果的精度。

例1.5：扩充上一个程序功能的例子。

$$t=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\cdots+\frac{1}{m}$$

编写 C 程序，计算 t 的值，其中 m 由键盘输入

C程序如下：

```
#include <stdio.h>
/*  函数声明    */
float func(int x);
/*          主函数          */
   main()
{  /*    说明部分，定义变量    */
   int m;
   float c;
      /*  提示串 */
   printf( "\n Please enter 1 integer number  :");
      /*    输入变量 m 的值    */
   scanf( "%d",&m);
     /*调用 func 函数，将得到的值赋给 c  */
   c = func(m);
     /*输出 c 值*/
   printf("\n the sum =  %f\n",c);
   return 0;
}
/*定义 func 函数，函数值为浮点型，形参 x 为整型数据 */
    float func(int x)
{  /*func 函数中的说明部分，定义本函数中用到的变量 i,sum */
   /*  i  为整型，sum 为浮点型，同时 sum 赋初值 1.0        */
   int i;
   float sum=1.0;
   for( i=2;i<=x;i++)
   sum = sum + 1.0/i;
       /*  将 sum 的值返回，通过 func 带回调用处    */
   return sum;
    }
```

程序分析：

1. 由功能划分确定函数划分本程序进一步表达了 C 语言基于函数的基本结构。上例中程序的执行过程是，首先在屏幕上显示提示串，请用户输入一个整数，回车后计算出累加和并在屏幕上显示。程序需要处理三件事情：从键盘接受输入 m ;计 算 累

加和（其功能是接受 main 传递的 m，计算出一加二分之一，加三分之一，一直加到 m 分之一的和，并把它返回给 main）；输出计算结果。所以程序除了一个主函数 main 以外，还包含计算累加和的函数 func，处理键盘输入的函数 scanf 和处理输出的函数 printf ，输入和输出这两个函数都是系统库函数，由 Turbo C 系统库提供，用户只要按规定使用而不必自己编写，但累加和计算函数 func 是一个由用户自己编写的自定义函数。

2. 键盘输入函数 scanf：本程序中的 scanf 语句的作用是从键盘上输入一个整型数给整型变量 m。%d 的含义与前面介绍的 printf 的用法一致，表示按十进制整型数输入，&m 的含义是将输入的数输入给变量 m，注意不要漏写“ &’,,其含义下一章将作详细介绍。

3. 自定义函数 func：

函数定义的一般形式为：

```
                    /* 首部 */
返回值类型    函数名（形参类型  形参一，形参类型  形参二，…）
{          /*  函数体  */
    说明部分
    执行部分
}
```

对照上面的例子：

```
返回值类型        函数名（形参类型  形参）              /* 首部 */
float       func      int     m    )
{                                 /*  函数体  */


}
```

一个函数名后面必须跟一对圆扩号，函数参数可以有也可以没有，例如前面介绍到的主函数 main ，没有用到参数，写成 main()（也可以使用参数,将在第 5 章中介绍它的用法）

函数体一般包括两部分：

说明部分：

在此定义函数内部所要使用到的变量，对照上面的例子，

```
int i;
float sum=1.0;
```

另外，对被调用的函数也需要进行说明。

执行部分：

由若干条语句组成。对照上面的例子，

```
for( i=2;i<=m;i++ )
    sum = sum + 1.0/i;
return sum
```

func 函数计算得到的值由 return 语句返回给 main 函数。关键字 return 可以后跟任何表达式，函数不一定都返回一个值。不含表达式的 return 语句用于控制返回调用者（但不返

回有用的值），如同在达到函数的终结右花括号时脱离函数一样。调用函数也可以不用一个函数所返回的值。本例在 main 函数末尾有一个 return 语句,由于 main 本身也是一个函数，它也可以向其调用者返回一个值，这个调用者就是程序的执行环境。一般而言，返回值为 0 表示正常返回，返回值非 0 则引发异常或错误终止条件。从简单考虑，大多数情况下 main 函数中都省去了 return 语句,如果需要向执行环境返回状态,就可以在 main 函数中用 return 语句返回运行状态。在某些情况下，函数体也可以既没有声明部分，也没有执行部分。

4. 如何调用函数 func：main 函数前的说明语句 float func(int m) 表明 func 是一个有一个 int 类型形参并返回一个 float 类型函数值的函数。这个说明叫做函数原型，要与函数的定义和使用相一致。 main 主函数通过语句 c = func(m) ;调 用 func 函数，调用时将实际参数 m 的值传送给 func 函数的形式参数 x ,通过执行 func 函数得到一个返回值，把这个值赋给变量 c。
5. 一个 C 程序的主函数 main 可以放在程序的任何位置，例如本程序的 main 也可以放在 func 函数后面，写成 下面的样子。不论 main 放在哪儿，程序都是从 main 开始执行。

```
#include <stdio.h>
float func(int x)
{  int i;
   float sum=1.0;
   for(i=2;i<=x;i++)
     sum = sum + 1.0/i;
   return sum;
}
main()
{  int m;
   float c;
   printf("\n Please enter 1 integer number:");
   scanf("%d",&m);
   c = func(m);
   printf("\n the sum =  %f",c);
   return 0;
}
```

通过以上几个例子的分析，说明简单的 C 程序的编写并不是十分的困难，从书写清晰，便于程序的阅读，理解和维护的角度出发，在一开始就要养成良好的书写程序习惯。

1. 虽然 C程序中，一行可以写多条语句，一条语句可以分写在多行，为清晰起见，一般一条语句单独占一行。
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。以便看起来层次更加清晰，增加程序的可读性。

4. 标识符，关键字之间必须至少加一个空格以示间隔。若已有明显的间隔符，虽然也可不再加空格来间隔，但也可以加一个空格来增加清晰度。

1.2 C 语言常用符号

要写文章，首先要学会写字，要编写 C 语言程序，首先要了解 C 程序中使用的符号。从上面的几个 C 程序例子可以看出，C 语言程序中，有些符号是 C 语言规定的符号，像 main、int、float、for、+ 、-、 * 等，有些是编程者自己使用的符号，像 sum、i、x 等。那么 C 语言规定了哪些符号？自己使用的符号又需要遵照什么样的规定？在 C 语言中使用的单词分为六类：标识符、关键字、常量、字符串字面值、运算符、分隔符。空格符、制表符、换行符、换页符和注解等统称为空白符。空白符在程序中仅起间隔作用，编译程序对它们忽略不计。因此在程序中使用空白符与否，对程序的编译不发生影响，但在程序中适当的地方使用空白符将增加程序的清晰性和可读性。

1.2.1 C 语言的关键字

在 C语言中规定了 34 个符号，它们具有特定含义，必须用小写字母，不能被作它用，称为关键字。

34 个关键字分列如下：

auto，char，double，extern，float，int，long，register，short，static，struc，typedef，union，unsigned，break，case，continue，default，do，else，for，goto，if，return 、switch，while，sizeof，asm，enum，signed，far，near，huge 、void。

其中前面已经用到过 float、int、for、return。

1.2.2 标识符

就像每个人有不同的名字一样，在 C 语言程序中，为了区别各个变量、各个函数、各种类型，都必须为它们取不同的名字。这些名字称为标识符。 C 语言规定，标识符以字母或下划线开头，后跟若干个字母、下划线或数字，大小写字母组成的标识符是不同的，标识符的长度没有限制。

例如以下标识符是合法的：

a、x、x3、BOOK_1、sum5。

以下标识符是非法的：

3s （以数字开头）、s*T（出现非法字符*）、-3x （以减号开头）、（bowy-1）出现非法字符-（减号）。

需要强调，C 语言中系统规定的标识符，例如 main、scanf、printf 等，在语法规定上允许用户改变它们原来的含义，但这样容易引起混淆，通常不把它们另作它用。

在使用标识符时还必须注意以下几点：

1. 标准 C 不限制标识符的长度，但它受到各种版本的 C 语言编译系统限制，同时也受到具体机器的限制。例如 MS C 规定标识符前八位有效，当两个标识符前八位相同时，则被认为是同一个标识符。 Turbo C 允许 32 个字符。
2. 标识符虽然可由程序员随意定义，但标识符是用于标识某个量的符号。因此，命名应尽量有相应的意义，以便于阅读理解，作到“见名知义”。
3. 在关键字，标识符之间必须要有一个以上的空格符作间隔，否则将会出现语法错误。例如把 int a;写成 inta;C 编译器会把 inta 当成一个标识符处理，其结果必然出错。

1.2.3 其他的符号

C 语言还规定了其他一些符号，例如运算符、分隔符等，相关内容将在以后的章节中介绍。

1.3 C 语言程序的上机调试步骤

C 语言是一种编译型的高级语言，描述解决问题算法的 C 语言源程序文件（*.c），必须先用 C 语言编译程序(compiler）将其编译，形成中间目标程序文件（*.obj），然后再用连接程序(linker）将该中间目标程序文件与有关的库文件（*.lib）和其他有关的中间目标程序文件连接起来，形成最终可以在操作系统平台上运行的二进制形式的可执行程序文件（*.exe）。所以从纸上写好的一个 C 语言源程序文字到可以在计算机操作系统平台上执行的可执行程序文件需要经过以下几个上机步骤，如图 1-1 所示。

1. 编 辑（Edit）：用任何一种编辑程序将源程序文字输入计算机，形成源程序文件，这期间必须注意严格按照 C 语言的语法规则，特别注意编辑程序是否添加了格式字符，千万不可出现额外不允许的特殊字符，例如全角的关键字。
2. 编 译（Compile）：将上一步形成的源程序文件作为编译程序的输入，进行编译。编译程序会自动分析、检查源程序文件的语法错误，并按两类错误类型（Waning、Error）报告出错行和原因。用户根据报告信息修改源程序，再编译，直到程序正确后，输出中间目标程序文件。
3. 连 接（Link）：使用连接程序，将上一步形成的中间目标文件与所指定的库文件和其他中间目标文件连接，这期间可能出现缺少库函数等连接错误，同样连接程序会报告出错误信息。用户根据错误报告信息再修改源程序，再编译，再连接，直到程序正确无误后输出可执行文件。
4. 运 行（Run）：上步完成后，就可以运行可执行文件，得到运行结果。当然也可能由于解决问题的算法问题而使源程序编写具有逻辑错误，得到错误的运行结果。或者由于语义上的错误，例如用 0 做除数，出现运行时错误（ Division by zero）. 这就需要检查算法问题，重新从编制源程序开始，直到运行结果正确。如何保证结果的正确性？需要设计出测试计划，进行全面、细致而艰苦的测试工作。

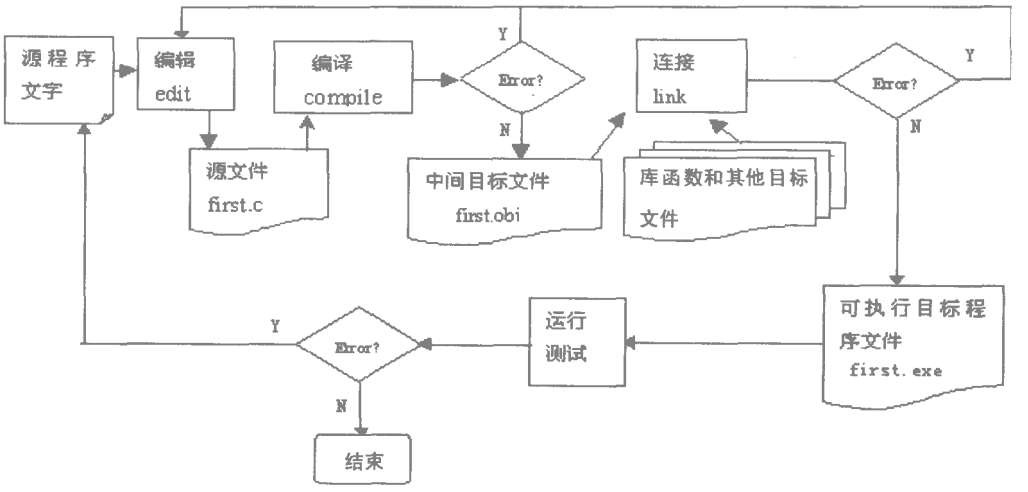


图 1-1 C 语言程序上机调试流程图

1.4 Turbo C 集成开发环境

Turbo C 是美国 Borland 公司生产的一套 DOS 平台上的 C 语言编译系统。Borland 公司是一家专门从事软件开发、研制的公司。该公司相继推出了一套在 DOS 平台上使用的 Turbo 系列软件,如 Turbo Basic、Turbo Pascal、Turbo Prolog 等，这些软件提供了一种新的集成化开发环境，通过一系列下拉式菜单选择将源文件编辑、编译、连接以及运行集成在一个窗口中，大大方便了程序的开发过程，很受用户的欢迎。另外 Turbo C 2.0 还在 ANSI C 的基础上增加了图形库和文本窗口函数库，大大满足图形处理的需要。并且在原来集成开发环境的基础上增加了查错功能；可以在 Tiny 模式下直接生成 .COM（数据、代码、堆栈处在同一 64KB 内存中）文件；可以对数学协处理器（支持 8087/80287/80387 等)进行仿真。本书采用 Turbo C 2.0 作为 C 语言程序的编程调试环境，在 Windows 98/2000 的 DOS 窗口下使用。

1.4.1 Turbo C 2.0 的安装

首先在磁盘上建立一个文件夹，例如 D:\TC2，然后将 Turbo C 2.0 所有程序安装（INSTALL）到该文件夹中，安装成功以后在该文件夹下分别装有 TC.EXE(集成开发平台)、(TCHELP.TCH)帮助文件、THELP.COM(读取 TCHELP.TCH 的驻留程序)、TCCONFIG.EXE(配置文件转换程序）、 MAKE.EXE(项目管理工具）、 TCC.EXE （命令行编译程序）、TLINK.EXE(Turbo C 系列连接器)、TLIB.EXE (Turbo C 系列库管理工具）等主要文件和子目录 LIB 和 INCLUDE , LIB 子目录中存放库文件，INCLUDE 子目录中存放所有头文件。所有程序安装成功以后，在桌面上建立快捷方式 TC2.0。

1.4.2 Turbo C 2.0 :集成化操作界面

双击 Win98/2000 下的快捷方式 TC2.0 图标，屏幕出现 Turbo C 2.0 集成化操作界面，如图 1-2 所示。

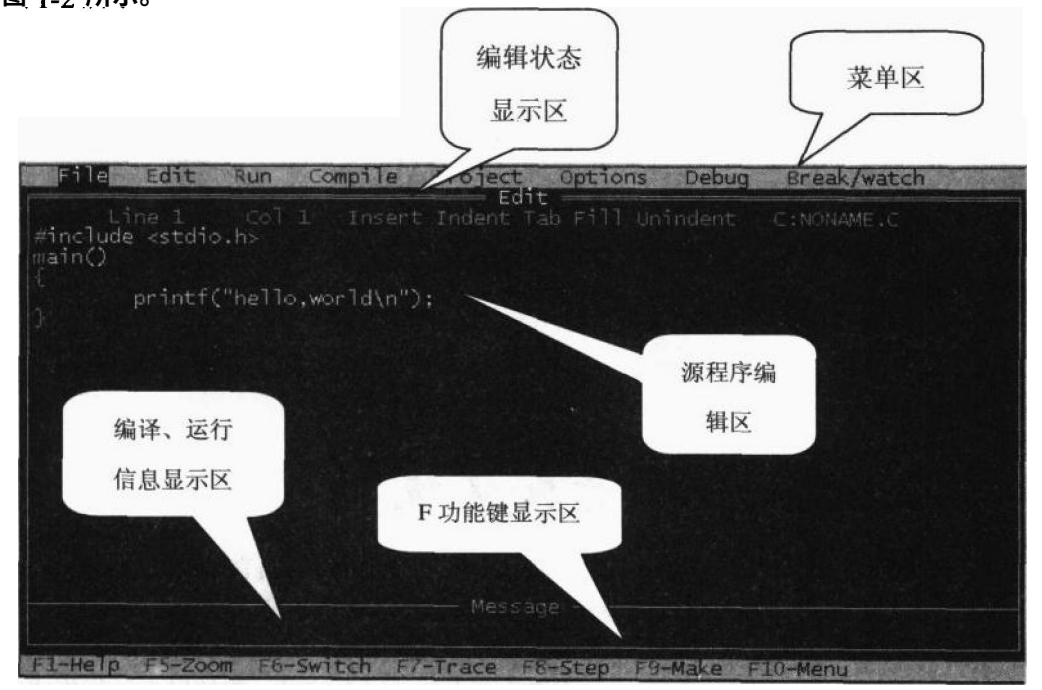


图 1-2 Turbo C 2.0 集成化操作界面

从图 1-2 可见，集成化操作界面包括五个主要区域：菜单区、编辑状态显示区、源程序编辑区、编译、运行信息显示区、F 功能键区。

1.4.3 Turbo C 2.0 简单操作

1、菜单操作

菜单是下拉菜单结构，具有八个菜单选择项，分别提供文件操作(File)、编辑操作(Edit)、运行操作(Run)、编译操作(Compile)、项目文件操作(Project)、选项操作(Options)、调试操作(Debug)，中断/观察操作(Break/Watch)等八个方面的选择。其主要操作方法与其他商用软件的下拉菜单操作一致：按键盘上的左、右箭头键或首字字母键使高亮度光条停在某一主菜单项上，此时该菜单项反白显示，表示选定该项，按回车键，出现该项的下拉子菜单项（除编辑菜单项外）。按键盘上的上、下箭头键或首字字母键使高亮度光条停留在某一子菜单项上，此时该子菜单项反白显示，表示选定该项。按下回车键启用该子项功能，另外大多数常用功能菜单也可以使用热键，例如装入文件（LOAD）可以直接按【F3】键，【Alt】+主菜单首字母可以打开主菜单项，【F6】键（Switch）可以进行编译、运行信息显示区和源程序编辑区的切换，【F10】键（Menu）可以进行源程序编辑区与菜单区的切换。

例1.6：打开一个已有的 C 源程序文件 first.c

主要操作过程：

1. 双击快捷方式 TC2.0，启 动 Turbo C 2.0。
2. 按【Alt+F】键，拉出文件操作下拉菜单。
3. 选择 Load 项，按回车键，此时屏幕出现一个对话框，如图 1-3 所示。表示要调入一个已有的源文件，请在对话框中输入文件名。
4. 直接在对话框中键入文件名 first.c （也可以只键入 first），按回车键。

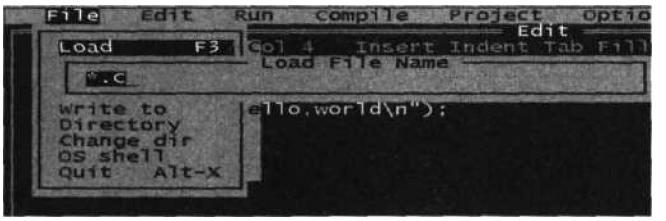


图 1-3 键入文件名调入一个以有的源文件

5. 也可以出现对话框后，直接按回车键，这时屏幕出现当前目录下所有的 C 源文件名（也可以在对话框中键入子目录名，选择 C 源文件的子目录），移动光标，停留在 first.c，按回车键，如图 1-4 所示。

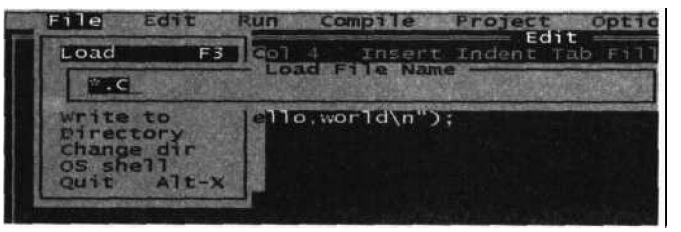


图 1-4 列出文件名选择调入一个已有的源文件

6. 进入编辑状态，光标停留在原来的位置。
7. 如果要编辑一个新文件，在 File 菜单下选择 New 项。这时 Turbo C 自动打开一个名为 NONAME.C 的新文件，光标停留在编辑区的第一行、第一列的位置。表示新编辑一个文件，给定的 NONAME.C 可以在保存文件时更改。
8. 在编辑过程中，注意按【F2】键经常保存文件，以免掉电或出现故障造成前功尽弃。

2、编辑操作

Turbo C 的编辑器是一个简单、方便的全屏幕编辑器。按【Alt+E】键启用编辑(Edit)菜单功能，进行源程序的输入或修改。编辑状态显示区显示当前光标所在的行号与列号、插入或覆盖状态、是否自动缩进方式，如图 1-5 所示。设置自动缩进方式，在回车以后，编辑器会自动将光标停留在与上一行相同的缩进位置上。一个优秀的编程人员往往使用自动缩进方式，以便使得程序十分的清晰明了。使用编辑功能键进行源文件的编辑，常用编辑功能键安排如表 1-1 所示。

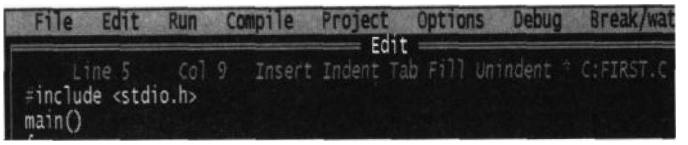


图 1-5 编辑状态显示区显示

表 1-1 编辑功能键

类别	功 能	默认键	类别	功 能	默认键
基本光标移动命令	字符左	Left	快速光标移动命令	行头	Home
	字符右	Right		行尾	End
	字左	Ctrl+A		窗口头	Ctrl+QE
	字右	Ctrl+F		窗口底	Ctrl+QX
	上行	Up		文件头	Ctrl+QR
	下行	Down		文件尾	Ctrl+QC
	上滚	Ctrl+W		块头	Ctrl+QB
	下滚	Ctrl+Z		块尾	Ctrl+QK
	上一页	PgUp			
	下一页	PgDn			
插入与删除命令	插入模式	Ins	块命令	标记块首	Ctrl+KB
	插入行	Ctrl+N		标记块尾	Ctrl+KK
	删除行	Ctrl+Y		复制块	Ctrl+KC
	删除至行尾	Ctrl+QY		删除块	Ctrl+KY
	删除光标左边字符	Backspace		块取消	Ctrl+KH
	删除光标处字符	Del		块移动	Ctrl+KV
	删除光标右边字符	Ctrl+T		读块	Ctrl+KR
				写块	Ctrl+KW
其他	异常结束操作	Ctrl+Break	其他	查找	Ctrl+QF
	制表位	Tab		查找并替换	Ctrl+QA
	自动缩进	Ctrl+OI		查找标记	Ctrl+QN
	定界符配对	Ctrl+Q[、Ctrl+Q]		退出编辑	Ctrl+KQ

C 语言程序中的许多定界符（{ }、[ ]、( ) 等）是配对使用的，在配对情况比较复杂时，如果能够自动查找出相配对的定界符，会给调试程序带来很大的方便。 Turbo C 的 Edit 具有这一功能，往往不被重视。通过查找配对定界符很容易找出下列程序（图 1-6）的错误在于 { 配对错。

主要操作过程：

1. 程序出现错误，语句很简单，但定界符配对不容易看清楚，将光标移动到第一个 { 位置。
2. 按【Ctrl+Q】键，光标停留在倒数第二行的 } 符号处（图 1-6），从错误信息马上可以看出错误原因在于 { 号的配对错。
3. 删除该 { 号。
4. 重新编译，马上通过。

图 1-6 配对定界符查找功能的使用

3、编译连接操作

按【Alt+C】键启用编译（Compile）菜单，可以拉出其下拉各子菜单项，包含编译子项 Compiler to OBJ)、连接子项(Link EXE File)、生成运行文件子项(Make EXE File)、建立所有文件子项(Build All)、初始 C 文件子项(Primary C File)等，如图 1-7 所示。先选择编译子项（Compiler to OBJ ）对当前编辑区的 C 源文件进行编译，根据调试信息显示区内的调试错误提示修改 C 源文件，正常后可以得到后缀为 OBJ 的中间目标文件。然后选择连接子项(Link EXE File)，将上一步得到的 OBJ 文件与相关的库文件或和其他的 OBJ 文件连接。同样调试信息显示区内会显示连接阶段出现的错误提示，如果不清楚引起的原因，可以在错误信息提示上按【F10】键，Turbo C 会显示可能引起错误的原因。根据提示修改 C 源文件，再按【F9】键(将编译和连接合并成一步)继续编译连接，没有错误后得到后缀为 EXE 的运行文件。

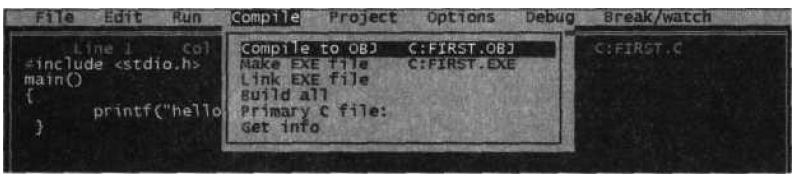


图 1-7 编译连接操作

例1.7：在编译和连接阶段出现的错误有致命错误（Error)、警告错(Warnings）。警告错仍然可以继续进行下一步的操作。Turbo C 不但能够把这些错误报告在编译、运行信息显示区，而且还能够将错误原因和错误位置大致地定位显示。这样可以在访问源代码的同时看到这些报告信息,给用户修改语法错误(编译时)和评估编译器给出的警告提供方便。也要注意有时所报告的行号并不是真正的出错行，需要结合上下文才能找出真正的出错行

现将上面的 first.c 制造几点语法错误，将第一行包含语句的 # 去掉，再去掉第五行 printf 语句中的后引号，将 printf 写成 print。错误的程序看上去是这样的：

```
include <stdio.h>
main()
{
     print("Hello, world\n);
}
```

主要操作过程：

1. 按【Alt+C】键，光标停留在(Compile to OBJ） 项上。
2. 按回车键，Turbo C 报告两个致命错误，如图 1-8 所示。

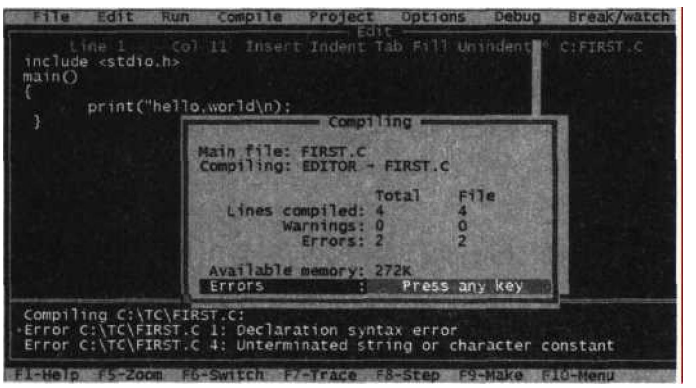


图 1-8 编译错误报告

3. 当看见编译窗口中的 Press any key 提示时，按任意键，消息窗口立刻被激活，亮条出现在第一个错误或警告上，这时编辑窗口中也会有一亮条，它标志着编译器给出的错误或警告在源代码中的相应位置。这时可用光标键将消息窗口中的亮条上下移动，注意到编辑窗口中的亮条也随着跟踪源代码中错误发生的位置。如果消息窗口太长看不见，可用左、右光标水平滚动消息，为了一次能够多看点报告信息，可按【F5】键放大信息窗口。放大后，编辑窗口不见了，因此不能进行错误跟踪。
4. 将信息窗口中的亮条置于第一个错误信息上，回车，光标移到编辑窗口中错误产生处，同时在编辑区第一行红色显示第一个错误信息报告。
5. 分析错误原因，（如果错误原因一时不易看出，可以按【F1】键，请教 HELP，按【Esc】键脱离 HELP)将第一行的＃补上。

6. 按【F6】键，光标回到信息窗口。移动光标使其停留在下一个错误报告信息上，按照同样方法修改错误，将字符串右双引号补上。
7. 另一种方法不用回到消息窗口，只要按【Alt+F8】键，键编译器就会将光标移至消息窗口中列的下一个错误。按【Alt+F7】键可移至前一个错误。
8. 按【Alt+C】键，重新编译，编译成功。

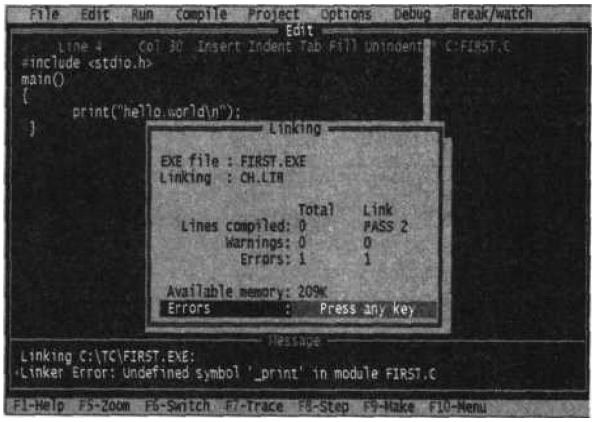


图 1-9　连接错误报告

9. 按【Alt+C】键，下移光条停留在 Link EXE file 选项上，按回车。开始连接操作，出现连接阶段错误报告，如图 1-9 所示。
10. 错误报告信息提示标识符 print 没有定义，实际是将 printf 误写成 print，改正后，重新编译连接，也可以直接按【F9】键，编译连接一步进行。
11. 编译连接成功。产生的可执行文件 first.exe，如图 1-10 所示。

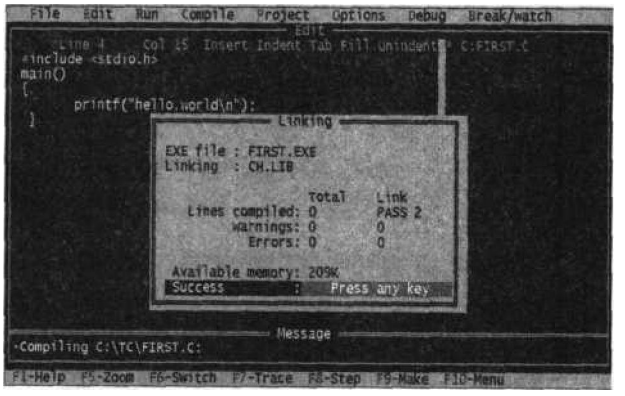


图 1-10　编译连接完成

12. 按【Ctrl+F9】键，运行成功生成的 first.exe，按【Alt + F5】键，切换到 DOS 用户界面，观看程序输出。