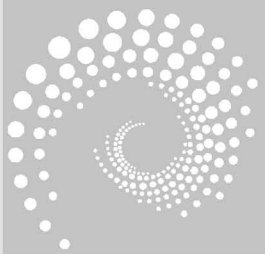
第1章

上机实验

CHAPTER 1



1.1　熟悉 C 语言编程环境

1.1.1　实验目的

（1）熟悉 C 语言编程环境 Dev-C++ 6.3，掌握运行 C 语言程序的基本步骤，包括编辑、编译、连接和运行。

（2）了解 C 语言程序的基本框架，能够编写简单的 C 语言程序。

（3）了解程序编译报错的含义，能找到并改正程序中的语法错误。

1.1.2　实验内容

1. 验证实验

【实验 1】 编写程序，在屏幕上显示“THIS IS A C PROGRAM.”。

源程序。

```
#include "stdio.h"
int main(void)
{
  printf("THIS IS A C PROGRAM.\n ");
  return 0;
}
```

下面在 Dev-C++ 6.3 的编程环境下，以上述 C 语言程序的源程序为例，介绍运行一个 C 语言程序的基本步骤。

（1）下载 Dev-C++ 6.3 软件并完成安装。下载地址为 https://sourceforge.net/projects/embarcadero-devcpp/。单击“Download”按钮，完成下载，如图 1-1 所示。“Embarcadero_Dev-Cpp_6.3_TDM-GCC_9.2_Setup.zip”软件解压后的安装操作和普通软件相同，请自行完成。

图 1-1　Dev-C++软件下载页面

（2）建立自己的文件夹。在计算机磁盘中新建一个用于存放 C 程序的文件夹，比如

D:\cpro。

(3) 启动 Dev-C++ 6.3。执行“开始”→所有程序→“Embarcadero Dev-C++”→“Dev-C++”(如图 1-2 所示),或者双击桌面的“Embarcadero Dev-C++”图标,进入编程环境。

图 1-2　Dev-C++ 6.3 软件打开方式

(4) 新建和编辑文件。在软件的初始界面的右侧单击“新文件”,然后直接在“未命名 1/Untitled1”文件里编辑代码。也可以依次单击“文件”→“新建”→“源代码”,或者直接使用快捷键 Ctrl+N 完成创建新文件的操作。

在“未命名 1”文件中第 1 行开始输入所有的程序代码。

主窗口(如图 1-3 所示)的顶部是主菜单栏,其中包括 10 个菜单项,即“文件”“编辑”“搜索”“运行”等。主窗口的左侧是项目管理工作区窗口,右侧是用于程序编辑的文件窗口,下面是调试信息窗口。项目管理工作区窗口显示所设定的项目的相关信息,程序编辑窗口用来输入和编辑源程序,调试信息窗口用来显示程序的出错信息,表示结果“无错误(Errors)”或“警告(Warnings)”。

此外,可以依次单击“工具”→“编辑器属性”,在“编辑器属性”对话框中的“显示”选项卡中,修改字体的大小,可以使代码方便观看。注意,修改字体属性之后,在新建的文件中才会执行这项修改。

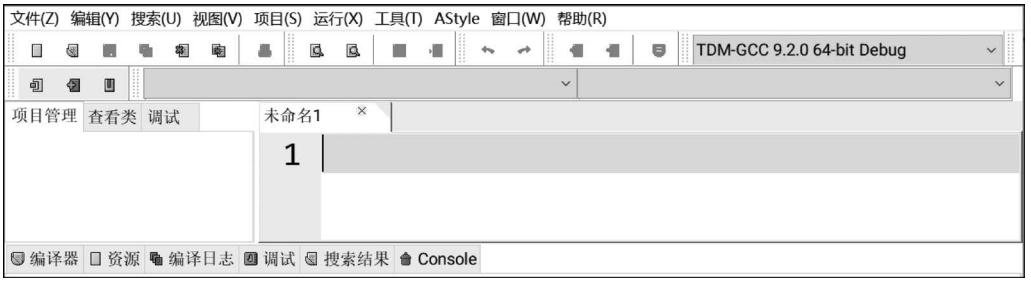


图 1-3　Dev-C++ 6.3 的主界面

(5) 保存文件。编辑代码后,依次单击“文件”→“保存”,或者直接使用快捷键 Ctrl+S,打开“保存为”对话框,如图 1-4 所示,选择保存位置(D:\cpro)、输入文件名称(test1)和选择保存类型(C source files (*.c)),完成保存新文件 D:\cpro\test1.c 的操作。

(6) 编译和连接文件。编译时,编译器首先要对源程序检查语法错误,当发现错误时就在屏幕上显示错误的位置和错误类型的信息。此时,要再次调用编辑器进行查错、修改。然后,重新编译,直至排除所有语法和语义错误。具体操作如下所述。

单击“运行”下拉菜单中的“编译”选项,或者直接使用快捷键 F9,如图 1-5 所示。经过短暂的编译处理时间,界面下方的“编译日志”子窗口会出现编译的相关信息,如图 1-6 所示。目前显示“错误和警告个数均为 0”,说明当前代码没有任何错误信息。

如果代码出现了语法错误,“编译”操作后会在界面下方“编译器”子窗口出现报错的提示信息,包括出现错误的文件名、报错的行号、列号和错误原因信息,具体改错操作可以查看后面“陷阱实验”的内容。

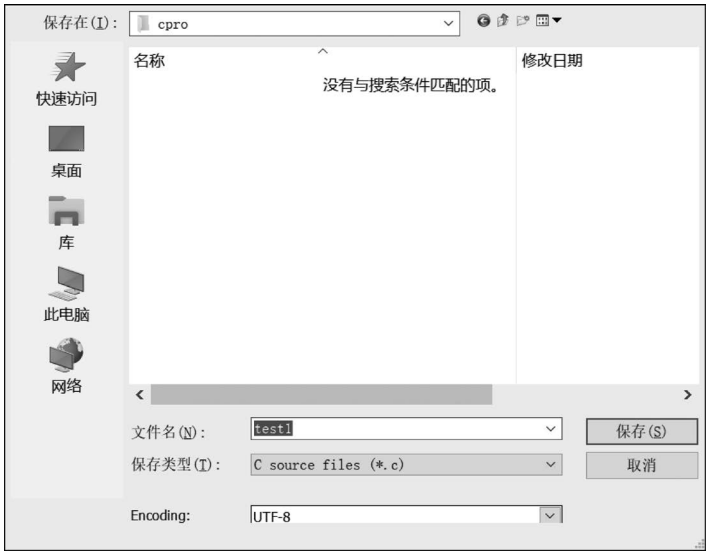


图 1-4　保存对话框

图 1-5　“运行”菜单

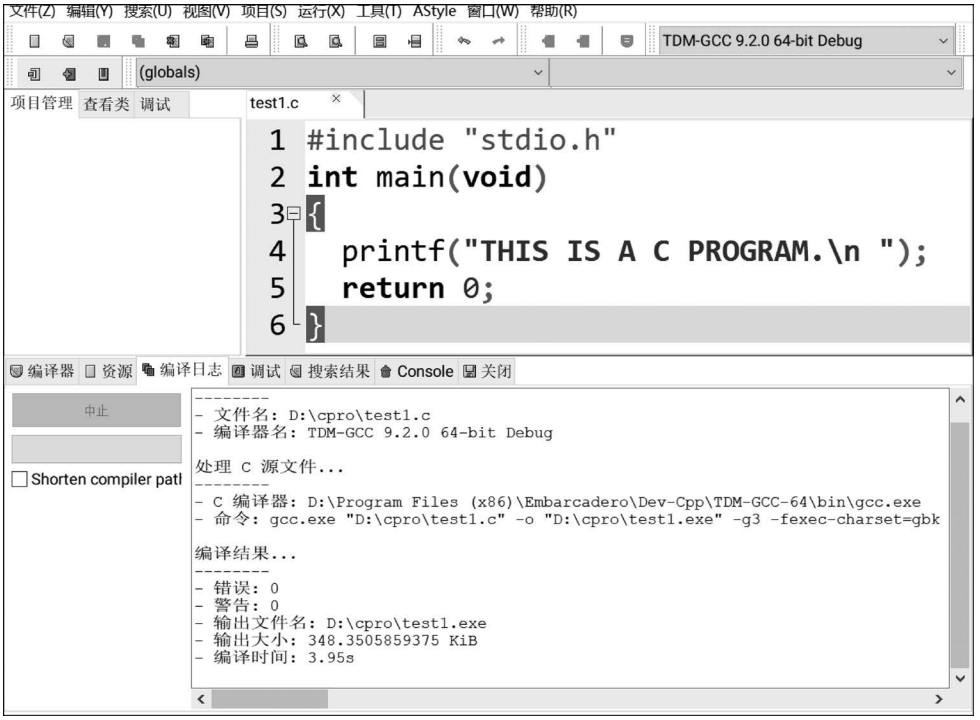


图 1-6　“编译日志”窗口信息

如果之前从未保存过该程序，在第一次单击“编译”选项后，屏幕上会出现一个“保存”对话框，按照要求选择保存的路径和输入文件名。编译完成后会生成一个可执行文件，并在下方编译日志中显示源程序编译后的结果信息。

编译后产生的目标程序还不能直接运行。连接过程就是目标文件和其他分别进行编译生成的目标程序模块及系统提供的库函数进行连接，生成可执行文件（文件扩展名是

“.exe”)。Dev-C++ 6.3 把“连接”步骤与“编译”步骤一并完成,因此在图 1-6 中的编译结果信息中有“输出文件名:D:\cpro\test1.exe”,说明可执行程序已经正常生成。

(7) 运行程序。程序通过了编译、连接生成执行文件后,就可以运行该程序了。操作方法为:单击“运行”菜单下的“运行”选项,或者直接使用快捷键 F10,如图 1-5 所示。

程序运行结果如图 1-7 所示。如果执行程序后得到了用户需要的结果,则 C 语言编写程序的过程结束。否则,要进一步检查源程序,重复编辑、编译、连接、运行,直到得到用户满意的结果为止。

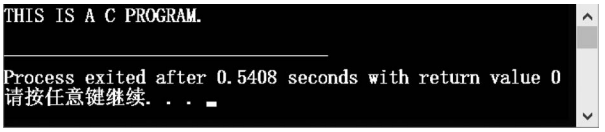


图 1-7　程序运行结果

(8) 算法描述方法。

在解决问题之前最好能给出对应解决问题的步骤,算法就是描述一个问题的具体步骤。现在给出一个简单的数学问题,要求用传统流程图、N-S 流程图、伪代码的方式来描述算法。

例如:求方程 $ax^2+bx+c=0$ 的根。分别考虑以下三种情况:①两个不等的实根;②两个相等的实根;③两个虚根。

第一步,必备知识。

普通的流程图、N-S 流程图和伪代码。

第二步,任务实施。

传统流程图由不同的图形组成,不同的几何图形表示各种类型的操作,如图 1-8 所示。在图形上用简明扼要的文字和符号表示具体的操作,并用带有箭头的流程线表示操作的先后次序。

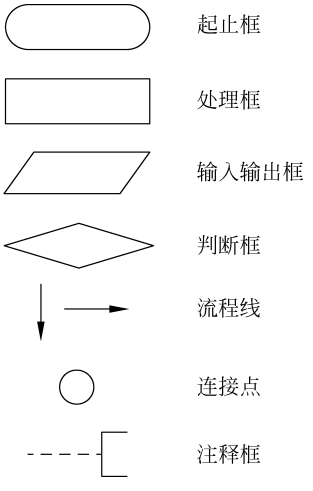


图 1-8　常用的传统流程图的基本符号

描述求方程 $ax^2+bx+c=0$ 根的问题中,首先使用起止框表示算法的开始。然后使用输入输出框,表示对数据 a,b,c 的输入操作。其次使用处理框,求 d 的值。由于方程需要根据 d 的不同取值判断根的情况,因此使用判断框 d>=0。当该判断成立时,再使用判断框 d==0,由此产生 3 种不同的情况,每种情况中均使用处理框计算 2 个根。然后,使用输入输出框将 2 个根输出。最后使用起止框表示结束。因此绘制方程根问题的传统流程图如图 1-9 所示。

(9) N-S 流程图。

N-S 流程图,又称为盒图,采用图形的方法描述处理过程,将全部算法写在一个大的矩形框中,框内包含若干基本处理框,没有指向箭头。N-S 图描述算法的优点是形象直观、可读性强,限制了随意的控制转移。

利用 N-S 图描述 3 种基本流程结构的形式,如图 1-10 所示。

根据传统流程图中对方程根问题的算法过程可以按输入、处理、判断、再判断、处理和输出步骤依次转换成 N-S 流程图,如图 1-11 所示。

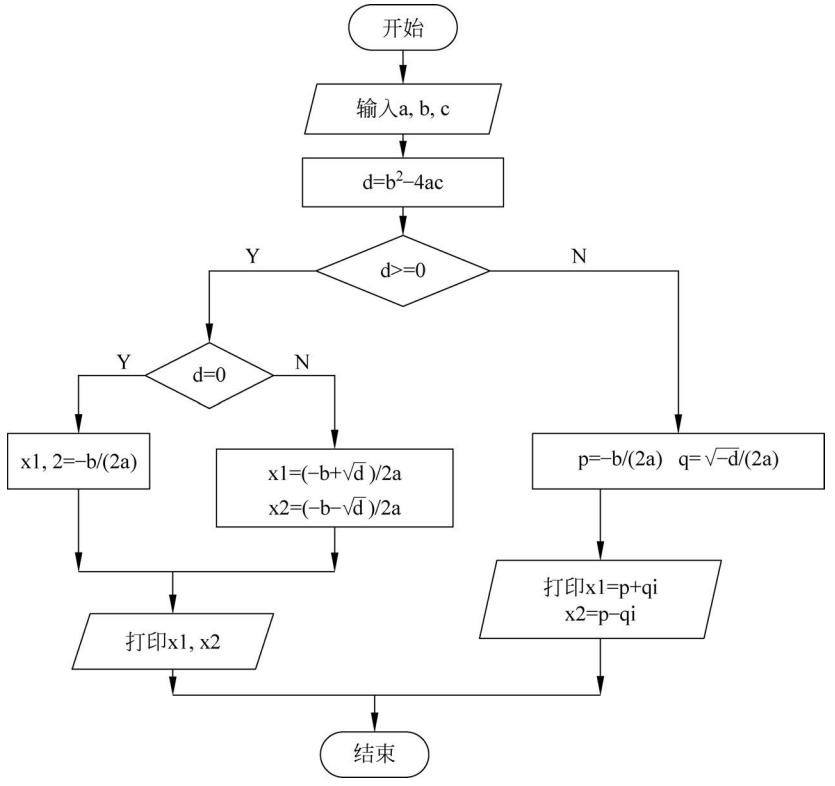


图 1-9　方程根问题传统流程图

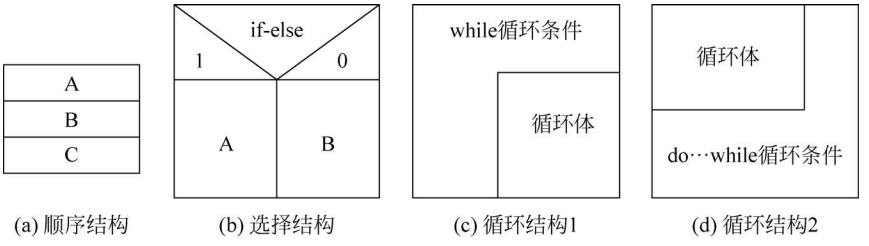


图 1-10　N-S 图描述 3 种基本流程结构

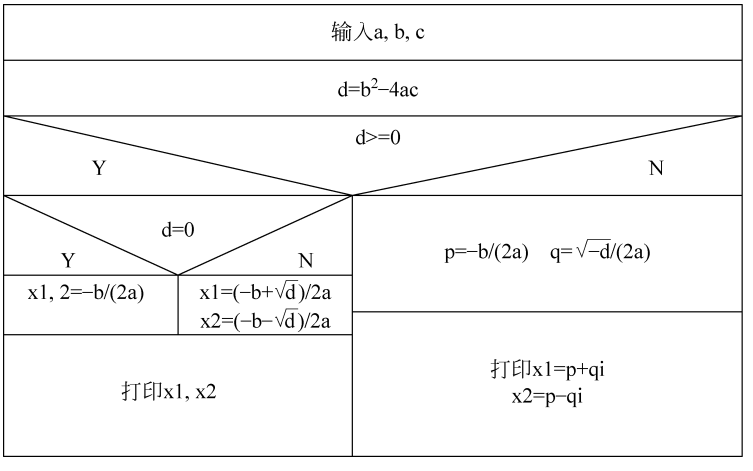


图 1-11　方程根问题 N-S 流程图

(10) 伪代码。

```
输入 a,b,c
d = b^2 - 4ac
if d >= 0
then
  if d = 0
  then
    x1,x2 = - b/(2a)
  else
    x1 = ( - b + √d/)/2a
    x2 = ( - b - √d/)/2a
  end if
  print x1,x2
else
    p = - b/(2a)
    q = √- d/(2a)
    print x1 =  p + qi, x2 = p - qi
end if
```

拓展思考。

(1) 能否在屏幕中输出中文信息?

(2) 如果输出的中文信息出现乱码应该如何处理?

(3) 使用 Dev-C++ 6.3 调试程序的步骤是什么?

2. 陷阱实验

【实验 1】 改正下列程序中的错误。在屏幕上显示“How do you do?”。

源程序(有错误的程序)。

```
1  #include <stdio.h>  // err1.c
2  int  mian(void)
3  {
4   printf(How do you do?\n")
5   return 0;              // 分号是中文输入
6  }
```

程序调试步骤如下。

(1) 编辑。按照以上代码编辑源程序 err1.c 文件。

(2) 编译。执行“运行”→“编译”,下方编译器信息窗口显示 11 条 error 错误和 warning 警告。

(3) 找出错误。观察编译器信息窗口中的每一条 error 或 warning 的具体提示信息,包括行号、列号、单元和信息,以此修正每个错误或警告。

(4) 首先,改正第 2 行代码中函数的名称。

改正错误 1。将第 2 行 mian 改为 main。

(5) 查看编译器信息窗口的第一条错误信息。双击该条出错信息,编辑窗口就会在代码行号处出现红色报错标志,指向程序出错的位置(如图 1-12 所示),一般在箭头的当前

行或者上一行，可以找到出错语句。这条出错信息的“行”显示“4”，“列”显示“10”，代表第 4 行第 10 列可能出错。当前光标所在的行号和列号可以通过窗口最下方的状态栏得知。

这条出错信息的“单元”显示“D:\err1.c”，表示本程序文件的位置与名称。“信息”显示“[Error] 'How' undeclared (first use in this function)”，表示“How”是一个未定义的变量，但此处“How”并不是变量，请仔细对照之前学习的代码格式，会发现出错的原因是“How”前少了前双引号。所以编译程序提示的出错信息只是对编译器从头至尾编译语句时发现了不符合语法规则的错误，给出相应错误提示信息，不一定是真实的错误原因。初学者应对照正确的代码不断练习，熟悉出错信息可能对应的出错原因。

改正错误 2。在第 4 行的“How”前加上前双引号。注意使用英文输入状态下的符号！

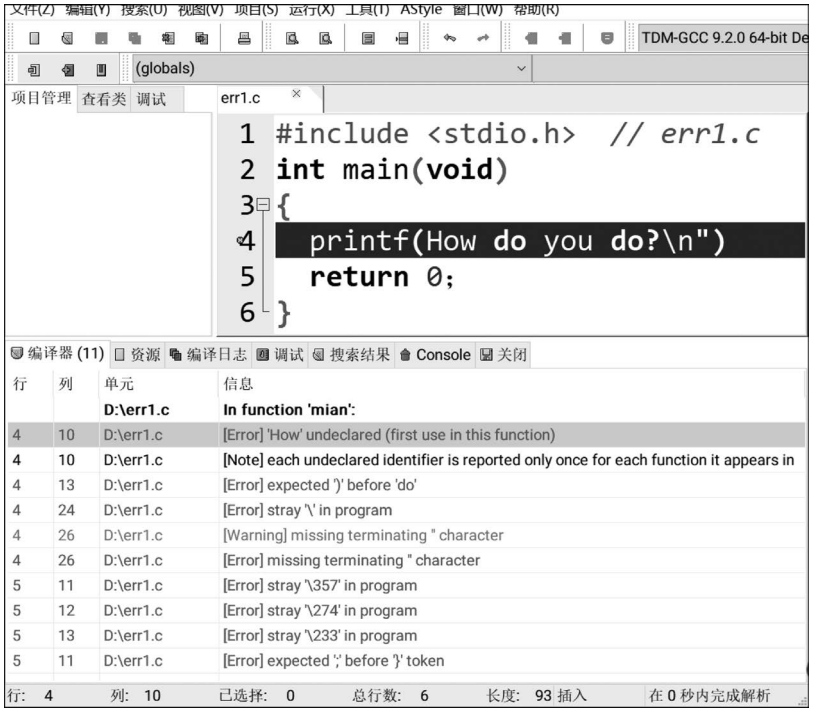


图 1-12　编译产生的错误信息

(6) 重新编译。下方编译器信息窗口显示 4 条 error。(如图 1-13 所示)修改了一个错误后，可以尝试重新编译，不建议要把所有错误修改之后再编译程序代码。

双击第一条出错信息，出错信息指出第 4 行第 29 列“expected ';' before 'return'”，即在“return”之前缺少语句的结束标志——分号。

改正错误 3。在第 4 行结尾添加分号。

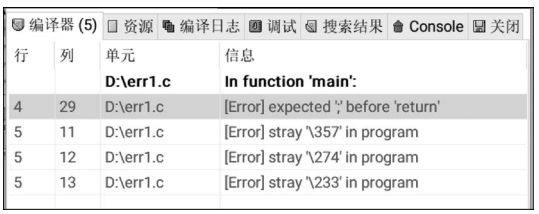


图 1-13　重新编译后产生错误的信息

(7) 重新编译。下方编译器信息窗口显示 4 条 error。(如图 1-14)本次出错信息指出第 5 行第 11、12、13 列出现了“stray '\357' in program'”类似错误,即在这些位置出现了中文输入状态下的全角符号而令编译器报错。

改正错误 4。将第 5 行的分号改为英文输入状态下的分号。

图 1-14　重新编译后产生错误的信息

(8) 重新编译、运行。

如果没有任何编译错误,执行“运行”→“运行”,则自动弹出运行窗口,显示运行结果,与题目要求的结果一致。按任意键自动关闭运行窗口。

运行结果:

```
How do you do?
```

【实验 2】 改正下列程序中的错误。在屏幕上显示以下 3 行信息。

```
****************
How do you do?
****************
```

源程序(有错误的程序)。

```
1  #include <stdio.h>
2  int  mian(void)
3 {
4  printf("****************\n");
5  printf"How do you do?\n")
6  printf("****************\n");
7  return 0;
8 }
```

对程序进行编译,“编译器”信息窗口显示错误信息。双击每个错误,观察源程序中的箭头位置所指行的代码,并分析错误原因,改错后重新编译,直至无错误信息并显示正确的运行结果。

错误行号:_____

错误原因:_____

正确语句:_____

运行结果如下。

```
****************
How do you do?
****************
```

3. 进阶实验

【实验 1】 在屏幕上显示一个短句 “what is a computer?”。

【实验 2】　在屏幕上显示下列图形。

```
   *
  ***
 *****
*******
```

【实验 3】　在屏幕上显示下列图形。

```
**********
*        *
*        *
**********
```

1.2　用 C 语言编写简单程序

1.2.1 实验目的

(1) 掌握算术表达式和赋值表达式的使用。
(2) 掌握基本输出函数的使用。

1.2.2　实验内容

1. 验证实验

【实验 1】　输入两个整数：100 和 50，求出它们的商和乘积，并输出。
第一步，必备知识。
数据类型、变量的定义、运算符、printf 函数和 scanf 函数。
第二步，描述算法。
(1) 定义四个整型变量；
(2) 输入两个整型数据；
(3) 求两数的商和乘积；
(4) 输出两数的商和乘积。
第三步，源程序。

```
1 #include "stdio.h"
2 int main(void)
3 { int a,b,c,d;
4   printf("\n 请输入两个整数:");
5   scanf("%d,%d",&a,&b);
6   c=a/b;
7   d=a*b;
8   printf("a/b=%d.\n",c);
9   printf("a*b=%d.\n",d);
10  return 0;
11 }
```

第四步，运行结果。

```
请输入两个整数:100,50↙
a/b = 2.
a * b = 5000.
```

拓展思考。

(1) 输入的两个整数之间是否可以没有逗号，或输入中文输入状态下的逗号？

(2) 如果两个整数相除的商不是整数值，运行程序是否能得到正确的运算结果？原因是什么？如输入为 100,40。

(3) 能否改编程序来计算两个实数的四则运算结果？

【实验 2】 把 150 分钟换算成用小时和分钟表示，然后输出。

第一步，必备知识。

数据类型、变量的定义、运算符、printf 函数和 scanf 函数。

第二步，描述算法。

(1) 定义分钟总数、小时数和分钟数三个整型变量；

(2) 根据题目的要求，输入分钟总数 150；

(3) 根据分钟总数求其对应的小时数和分钟数；

(4) 输出分钟总数转换的结果。

第三步，源程序。

```
1 #include "stdio.h"
2 int main(void)
3 { int time,hour,minute;
4   printf("\n 请输入分钟总数:");
5   scanf("%d",&time);
6   hour = time/60;
7   minute = time%60;
8   printf(" = %d 小时",hour);
9   printf("%d 分钟\n",minute);
10  return 0;
11 }
```

第四步，运行结果。

```
请输入分钟总数:150↙
 = 2 小时 30 分钟
```

拓展思考。

改写程序，当输入 x 小时 y 分钟时，程序将其换算成总分钟数。

2. 陷阱实验

【实验 1】 改正下列程序中的错误，求圆的周长。计算公式如下。

$$c = 2\pi r$$

其中，c 表示周长，r 表示面积。

源程序(有错误的程序)。

```
1  #include <stdoi.h>
2  #definePI 3.14
```

```
3  int  main(void)
4  {
5    int  r;c;
6    r = 3;
7    c = 2 * PI * r;
8    printf("r = d,c = %d\n",r,c);
9    return 0;
10 }
```

程序调试步骤如下。

(1) 编辑。按照以上代码编辑源程序文件。

(2) 编译。执行“运行”→“编译”,下方编译器信息窗口显示 1 条错误。

(3) 这个错误是第 1 行 11 列,代码中包含文件的位置。错误信息是“[Error] stdoi. h: No such file or directory”,指 stdoi. h 文件不存在。

改正错误 1。将第 1 行的 stdoi 改为 stdio。

(4) 重新编译。下方编译器信息窗口显示 10 条错误。修改了刚才第一个错误后,错误数量反而增加了,注意这不一定代表刚才的错误修改不到位,仔细查看错误提示信息,耐心修改程序中的每一条错误。

双击此时的第一条出错信息,出错信息指出第 5 行第 10 列“[Error] 'c' undeclared (first use in this function)”,即 c 未被定义(第一次在这个函数中被使用)。查看这一行,c 是需要被定义的变量,第 5 行的写法不能同时定义两个变量。

改正错误 2。将第 5 行改为“int r,c”。

(5) 重新编译、运行。如果没有任何编译错误,执行“运行”→“运行”,则自动弹出运行窗口,正常显示运行结果“r=d,c=3”,但是与题目要求的运行结果“r=3,c=18.840000”不一致。

这种情况以后会经常遇到,语法错误由编译器(Dev-C++)帮助编程者查找出来,但是类似现在这种逻辑错误就需要编程者仔细阅读题目要求和程序代码,分析目前的运行结果,思考并继续修改代码。

(6) 分析逻辑错误 1。分析目前的运行结果“r=d,c=3”,需要结合输出语句来查看“printf("r=d,c=%d\n",r,c);”。可以发现,“r=d,c=”这部分和 printf 中双引号内的内容对应,结合教材第 4 章中对 printf 函数的描述可知,“r=”之后没有出现数字 3,是因为“格式字符%d”漏写了百分号。

改正错误 3。在第 8 行的 d 前增加“%”,整行改为“printf("r=%d,c=%d\n",r,c);”。

(7) 重新编译、运行。

无编译错误,正常显示运行结果“r=3,c=18”,这与题目要求的结果“r=3,c=18.840000”仍不一致,但是已经成功一半了。

分析当前的逻辑错误,就集中在 c 的取值上,为什么结果不是小数呢? 结合配套的《C 语言程序设计》教材第 4 章中对 printf 函数的描述可知,格式字符%d 用来输出整数,因此 c 不可能输出题目需要的小数。

那么,如何修改使得运行结果完全正确呢?

(8) 分析逻辑错误 2。第 1 种思路,修改 printf 中输出 c 的格式字符为%f,修改编译运

行后发现运行结果仍然不正确。第 2 种思路，修改变量 c 的类型为实型，因为如果 c 是整型，它的取值一定是整数。

改正错误 4。在第 5 行将 c 的类型改为 float，“int r；float c；”，将第 8 行 c 的格式字符改为“%f”。

(9) 重新编译、运行。

运行结果(改正后程序的运行结果)。

```
r = 3,c = 18.840000
```

【实验 2】 改正下列程序中的错误。

计算某个数 x 的 2 倍并赋值给 y，最后分别以“y＝2 * x”和“2 * x＝y”的形式输出。

源程序(有错误的程序)。

```
1  #include < stdio.h >
2  int  main(void)
3  {
4    int  x,y;
5    y = 2 * x;
6    printf(" % d = 2 * % d",x);
7    printf("2 * d = % d",y);
8    return 0;
9  }
```

对程序进行编译，“编译器”信息窗口显示错误信息。双击每个错误，观察源程序中的箭头位置所指行的代码，并分析错误原因，改错后重新编译，直至无错误信息并显示正确的运行结果。

错误行号：_____

错误原因：_____

正确语句：_____

运行结果(改正后程序的运行结果)如下。

(假设输入 x 的值是 4)

```
8 = 2 * 4
2 * 4 = 8
```

3. 进阶实验

【实验 1】 已知某位学生的数学、英语和计算机课程的成绩分别是 87 分、72 分和 93 分，求该学生 3 门课程成绩的平均分。

输入、输出示例如下。

```
数学 = 87,英语 = 72,计算机 = 93,
平均分 = 84
```

【实验 2】 当 n 为 28 时，分别求出 n 的个位数字和十位数字。

输出示例如下。

```
整数 28 的个位数字是 8,十位数字是 2
```

【实验 3】 求一个数值的平方和立方。

输入、输出示例如下。

```
5↙
5 的平方是 25,5 的立方是 125
```

1.3　顺序结构程序设计

1.3.1　实验目的

(1) 能够编程实现简单的数据输入、输出和运算处理。

(2) 能够理解编译错误信息的含义，掌握简单 C 语言程序的查错方法。

1.3.2　实验内容

1. 验证实验

【实验 1】　将原文中的 3 个字符进行加密操作。加密转换规律是：按字典顺序，将原来字母后面的第 5 个字母替换原来的字母。例如要将“abc”译成密码，则“a”后面第 5 个字母是“f”，用“f”代替“a”，b 和 c 也以此类推，因为，“abc”应译为“fgh”。

第一步，必备知识。

变量定义，运算符。

第二步，描述算法。

(1) 定义三个字符数据；

(2) 输入这三个字符数据；

(3) 进行加密操作：给每个字符都加 5，并赋给它本身；

(4) 输出这三个字符数据。

第三步，源程序。

```
#include "stdio.h"
int  main()
{ char c1,c2,c3;
   printf("输入三个字符变量:");
   scanf("%c%c%c",&c1,&c2,&c3);
   c1 += 5;
   c2 += 5;
   c3 += 5;
   printf("译成密码是:%c%c%c\n",c1,c2,c3);
  return 0;
}
```

第四步，运行结果。

```
输入三个字符变量:abc↙
译成密码是:fgh
```

拓展思考。

(1) 如果输入三个字符间加入空格，思考产生的运行结果并分析其原因。提示：可以

对照 ASCII 表来寻找线索。

运行结果。

```
输入三个变量:a b c↙
译成密码是:f % g
```

(2) 如果修改输入函数语句为“scanf("%c,%c,%c",&c1,&c2,&c3);”,此时应该如何输入 3 个字符数据并得到正确结果?

【实验 2】 输入两个数据,它们分别为整型和浮点型。将整型数据转换成浮点类型数据;将浮点型转换为整型数据。

第一步,必备知识。

变量定义,强制类型转换运算符。

第二步,描述算法。

(1) 定义整型数据和浮点型数据;

(2) 输入这两个数,使用 scanf 或直接赋值;

(3) 将整型数据转换成浮点类型数据,并直接输出;

(4) 将另一个浮点型转换为整型数据,并直接输出。

第三步,源程序。

```
1  #include "stdio.h"
2  int  main()
3  {
4  int  n;
5  float   f;
6  n = 10;
7  f = 25.4;
8  printf("原始的数据为:n = %d,f = %f\n",n,f);
9  printf("(float)n = %f\n",(float)n);
10 printf("(int)f = %d\n",(int)f);
11 return 0;
12 }
```

第四步,运行结果。

```
原始的数据为:n = 10,f = 25.400000
(float)n = 10.000000
(int)f = 25
```

拓展思考。

(1) 如何验证强制类型转换运算是将浮点数直接取整还是将小数部分四舍五入?

(2) 强制类型转换运算一般出现在哪些情况中?

2. 陷阱实验

【实验 1】 改正下列程序中的错误。该程序将输入的两个小写字母转换成大写字母并输出。

源程序(有错误的程序)。

```
1  #include  "stdio.h"
2  int  main(void)
```

```
3  {
4    char a,b;
5    printf("请输入两个小写字母:");
6    scan("%c,%c",&a,b);
7    a=a+32;
8    b=b-32;
9    printf("%c %c",a,b);
10   return 0;
11 }
```

程序调试步骤如下。

(1) 编辑。按照以上代码内容编辑源程序文件。

(2) 编译。执行“运行”→“编译”,下方编译器信息窗口显示1条错误和1个警告。

(3) 查看第一个警告。这个警告指向第6行代码,信息提示“[Warning] implicit declaration of function 'scan'; did you mean 'scanf'? [-Wimplicit-function-declaration]”,指未对函数“scan”做声明,你是否想写“scanf”函数。

改正错误1。第6行的“scan”改为“scanf”。

(4) 重新编译和分析逻辑错误1。如果没有任何编译错误,执行“运行”→“运行”,则自动弹出运行窗口。按照“scanf”语句和题目要求,输入两个小写字母并且用逗号分隔。

运行结果如下。

```
请输入两个小写字母:a,b↙
```

可以观察到,没有输出结果,这说明发生了逻辑错误。

(5) 考虑输入是否正确,仔细观察“scanf”语句,发现“b”的前面漏写了“&”符号。

改正错误2。第6行的“b”变量前添加“&”。

(6) 重新编译、运行。发现运行结果与题目要求部分相符,代码仍然存在逻辑错误。

运行结果如下。

```
请输入两个小写字母:a,b↙
?B
```

(7) 分析逻辑错误2。“a”为何不能正确转换为大写字母“A”,而“b”字符却可以正确转换呢?观察代码第7和8行,发现了不同点。根据ASCII表的规律,小写字母要转换为对应大写字母需要减32。因此第8行是正确的。

改正错误3。第7行加号改为减号,即“a=a-32;”。

(8) 重新编译、运行。

运行结果(改正后程序的运行结果)如下。

```
请输入两个小写字母:a,b↙
A B
```

【实验2】 改正下列程序中的错误,求华氏温度100度对应的摄氏温度(c表示摄氏温度;f表示华氏温度)。计算公式如下:

$$c=5\times(f-32)/9$$

源程序(有错误的程序)如下。

```
#include <stdio.h>
int  main(void)
{  float   c,f;
   f = 100.0;
   c = 5 (f - 32)\9;
   printf("f = %.1f,c = %.1f\n",f,c);
   return 0;
}
```

对程序进行编译,“编译器”信息窗口显示错误信息。双击每个错误,观察源程序中的箭头位置所指行的代码,并分析错误原因,改错后重新编译直至无错误信息并正确显示运行结果。

错误行号:_____

错误原因:_____

正确语句:_____

运行结果(改正后程序的运行结果)如下。

```
f = 100.0 c = 37.8
```

3. 进阶实验

【实验 1】 求摄氏温度 26℃对应的华氏温度。计算公式如下。

$$f=(9\times c/5)+32$$

其中,c 表示摄氏温度;f 表示华氏温度。

输出示例如下。

```
c = 26   f = 78
```

【实验 2】 连续输入三个大写字母,输出各自对应的小写字母,输出字母之间有一个空格。

输入输出示例如下。

```
ABC↙
对应的小写字母是:a b c
```

【实验 3】 计算一个人的体质指数(BMI)。计算公式如下。

$$BMI=\text{体重}\div\text{身高}^2$$。(体重单位:千克;身高单位:米)

输入、输出示例如下。

```
55 1.6↙
BMI = 21.5
```

1.4　选择结构程序设计

1.4.1　实验目的

(1) 使用 if 语句计算分段函数。

(2) 正确书写关系表达式。

(3) 掌握基本输入函数的使用，能正确调用 C 语言提供的数学函数。

1.4.2　实验内容

1. 验证实验

【实验 1】 输入 x，计算并输出下列分段函数 f(x)的值。

$$y=f(x)=\begin{cases}2x, & x\neq 0\\ 0, & x=0\end{cases}$$

第一步，必备知识。

if 语句的双分支结构形式。

第二步，描述算法。

判断条件 x 的值是否为零。如果 x 的值不等于零，则 y 的值为 2x；否则为零。

第三步，源程序。

```
1   #include <stdio.h>
2   int  main(void)
3   {
4    int  x,y;
5    printf("Enter x:");
6    scanf("%d",&x);
7    if(x!=0){ y=2*x;}
8       else  { y=0;}
9    printf("f(%d)=%d\n",x,y);
10   return 0;
11   }
```

运行结果 1 如下。

```
Enter x:10
f(10)=20
```

运行结果 2 如下。

```
Enter x:0
f(0)=0
```

拓展思考。

(1) 将程序中 y=2*x;和 y=0;换下位置，如何修改条件表达式？

(2) 如果用单分支结构替换双分支结构，如何修改程序？

【实验 2】 写一个程序，输入 x，输出 y。

$$y=\begin{cases}x, & x<1\\ 2x-1, & 1<=x<10\\ 3x-11, & x>=10\end{cases}$$

第一步，必备知识。

if-else 多分支选择结构，if 语句的嵌套。

第二步，描述算法。

这是一个分段函数，当 x 取值不同时，y 的值也会发生变化。当 x＜1 时，y＝x；当 1＜＝x＜10 时，y＝2x－1；当 x＞＝10 时，y＝3x－11。这里使用 if 语句的嵌套完成。

第三步，源程序。

```
#include "stdio.h"
  int  main()
  {
   int  x,y;
   printf("请输入 x 的值:");
   scanf("%d",&x);
   if(x<1)
      {y=x;
      printf("当 x=%d 时,则 y=x=%d\n",x,y);}
    else  if(x<10)
          {y=2*x-1;
           printf("当 x=%d 时, 则 y=2*x-1=%d\n",x,y);
           }
       else
          {y=3*x-11;
          printf("当 x=%d 时, y=3*x-11=%d\n",x,y);
           }
  return 0;
  }
```

第四步，运行结果（运行三次，每一种情况都进行了测试）如下。

```
请输入 x 的值:-5↙
当 x=-5 时,则 y=x=-5
请输入 x 的值:5↙
当 x=5 时,则 y=2*x-1=9
请输入 x 的值:15↙
当 x=15 时,则 y=3*x-11=34
```

拓展思考。

（1）如果不用 if 的嵌套结构，只用三个简单的 if 单分支结构能否解决？

（2）如果用 switch 语句是否可以解决？比较 switch 语句和 if 嵌套结构的优缺点，什么情况下用 switch 语句算法更优？

【实验 3】 有 4 个圆塔，圆心分别为（2，2）、（－2，2）、（2，－2）、（－2，－2），圆半径为 1，如图 1-15 所示。这 4 个塔的高度分别为 10m。塔以外无建筑物。现输入任一点的坐标，求该点的建筑高度（塔外的高度为零）。

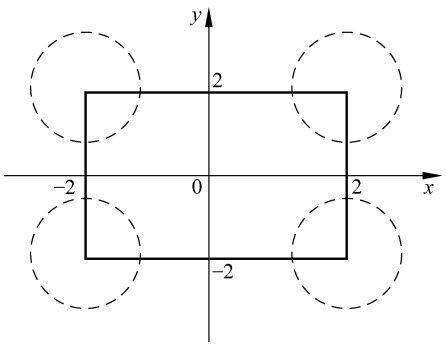


图 1-15　圆塔分布图

第一步，必备知识。

选择结构，if 语句。

第二步，描述算法。

（1）顺序结构，求各个塔点的距离；

（2）选择结构，判断是否在塔外。

第三步，算法分析如下。

(1) 输入某点坐标(x,y)。

(2) 求(x,y)到各塔心的距离 d1、d2、d3、d4。

```
d1 = (x - x1) * (x - x1) + (y - y1) * (y - y1);
d2 = (x - x2) * (x - x2) + (y + y2) * (y + y2);
d3 = (x + x3) * (x + x3) + (y - y3) * (y - y3);
d4 = (x + x4) * (x + x4) + (y + y4) * (y + y4);
```

(3) 判断是否在塔外，如果为真，(x,y)处高度为 0，(x,y)处高度为 10。

```
if(d1 > 1&&d2 > 1&&d3 > 1&&d4 > 1)h = 0;
printf("该点高度为 % d",h);
```

(4) 最后输出结果。

第四步，源程序如下。

```
1  # include "stdio.h"
2  int  main()
3  {
4   int  h = 10;
5   float   x1 = 2,y1 = 2,x2 = - 2,y2 = 2,x3 = - 2,y3 = - 2,x4 = 2,y4 = - 2,x,y,d1,d2,d3,d4;
6   printf("请输入一个点(x,y):");
7   scanf(" % f, % f",&x,&y);
8   d1 = (x - x1) * (x - x1) + (y - y1) * (y - y1);
9   d2 = (x - x2) * (x - x2) + (y + y2) * (y + y2);
10  d3 = (x + x3) * (x + x3) + (y - y3) * (y - y3);
11  d4 = (x + x4) * (x + x4) + (y + y4) * (y + y4);
12  if(d1 > 1&&d2 > 1&&d3 > 1&&d4 > 1) h = 0;
13  printf("该点高度为 % d",h);
14  printf("\n");
15  return 0;
16 }
```

第五步，运行结果如下。

```
请输入一个点(x,y):0.6,0.8↙
该点高度为 0
```

拓展思考如下。

测试该程序的其他全部情况。

2. 陷阱实验

【实验 1】 改正下列程序中的错误。输入实数 x，计算并输出 f(x)的值。

$$y=\ f(x)=\begin{cases}2x, & x=10\\ x, & x\neq 10\end{cases}$$

源程序(有错误的程序)如下。

```
1 # include < stdio.h >
2 int  main(void)
3 {
4  int  x,y;
5  printf("Enter x:");
```