第 3 章　设计一个程序

先行案例：简单程序设计

先来看一个问题。

例 3-1　从键盘上输入一个大写字母，输出其对应的小写字母。

对于程序设计过程而言，比软件设计过程简单，为了解决上述问题，可以分以下几步进行。

第一步，问题描述。对于本例子来讲，所要解决的问题是，如何将一个大写字母转化为小写字母。

第二步，程序中注意问题，输入与输出。例子中的输入即一个大写字母，输出是对应的小写字母。

第三步，算法描述。可以用简单的语言或者伪代码来描述算法，也可以画出程序流程图来表示，下面对问题进行分步骤简单描述。

（1）输入一个大写字符；

（2）获取这个大写字符的 ASCII 码值，并将这个码值加上 32，得到对应的小写字符的 ASCII 码值；

（3）根据得到的小写字符的 ASCII 码值输出对应的小写字符。

第四步，编码实现。

根据上述分析，采用 C 语言实现程序代码如下：

```
1  #include <stdio.h>
2  int main()
3  {
4      char ch;
5      printf("请输入一个大写字母: \n");
6      scanf("%c",&ch);
7      ch = ch + 32;
8      printf("该大写字母对应的小写字母是: %c\n",ch);
9      system("pause");
10 }
```

第五步，调试与排错。按照附录中阐述的 C 语言调试方法，对上述代码进行调试，运行，得到的结果如图 3-1 所示。

至此，整个问题得到解决。如果代码编写过程中有错误，则需要反复调试，才能够得到正确的结果。问题描述和算法在第 1 章和第 2 章中已经学习过，接下来要学习代码实现部分。例如，char 是 C 语言的一个关键字，但是为什么用 char，而不是用 int？还有没有其他的写法？这些问题将在本章逐步解决。

图 3-1　例 3-1 的运行结果

3.1　标识符与数据类型

到本章为止，我们见识了用来解决问题的一些 C 语言程序代码，在这些代码中，除了关键字、运算符和标点符号以外，剩下的是大量的由关键字定义的符号，统称为“标识符”。

例如，在例 3-1 中第 1 行的 stdio.h，第 2 行的 main，第 4 行的 ch，第 5 行的 printf，第 6 行的 scanf 等，这些都是标识符。

3.1.1　标识符的类型

就像每个人都要有一个名字，每个事物都要有一个名称一样，程序代码中用标识符来表示程序中需要用到的变量、常量、函数、语句块等。在例 3-1 中 stdio.h 表示头文件名称；而 main、printf 和 scanf 则是代表了函数名；ch 是一个变量名。在简单的结构化程序设计语言中，标识符一般有如图 3-2 所示的类型。

1. 常量与变量

本书不只在一个地方提到，从一定角度来讲，程序设计＝数据结构＋算法。如图 3-3 所示，数据在计算机中存储在存储器的一个个存储单元中，可以给用来存储数据存储单元起一个名字，即用一个标识符来表示这个存储单元。如果我们规定了某个存储好的数据或即将存储的数据不能更改，那么这个数据或者这个存储单元的名称就是一个常量。如果这个存储单元中的数据可以更改，那么这个存储单元的名称就是一个变量。

图 3-2　常见的标识符类型

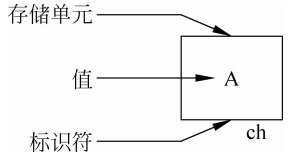

图 3-3　常量与变量

例如图 3-3 中的标识符 ch，它代表了一个存储单元，这个存储单元中放置了字符'A'，当然对于例 3-1 中的程序而言，这个存储单元也可以放'B'、'C'、'D'、…字符，因此标识符 ch 是一个变量。而'A'则是一个常量。

2. 函数

需要注意的是，程序中函数的概念不同于以往学习的数学中函数的概念。

在程序设计中，函数主要有两个方面的作用：第一是当一个程序较大时，可以分为若

干个小程序块，每一个模块用来实现一个特定的功能，这样有助于程序的实现和分工。第二是当一个功能需要多次使用时，可以设计成函数，以方便反复调用。在C语言中，一个主函数和若干个函数构成子程序，主函数调用其他函数，其他函数也可以互相调用，同一个函数可以被一个或多个函数调用任意多次。例3-1中的main、printf和scanf就是函数名。

3. 程序块与文件名

在程序设计中，常将一些常用的功能模块编写成函数，放在函数库中供公共选用。要善于利用函数，以减少重复编写程序段的工作量。函数库可以以文件的形式存放在固定的文件夹中，例3-1中的stdio.h就是编写好的函数库文件，在程序设计时，只需要将这样的文件引入到程序中，就可以使用其中的函数。stdio.h文件中，包括了printf和scanf两个函数的定义和说明，因此，在程序中不需要对这两个函数进行再次定义，也可以使用这两个函数来实现一定的功能。

3.1.2 标识符的命名

不同的计算机语言标识符有不同的命名规则，一般来讲，该语言默认的关键字是不能作为标识符出现的。对于C语言来讲，标识符只能是字母(A～Z、a～z)、数字(0～9)、下划线(_)组成的字符串，并且其第一个字符必须是字母或下画线。

例3-2 以下标识符是合法的。

b，xA，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)

注意：C语言区分大小写，“xA”和“xa”是两个不同标识符。

正如程序的注释一样，标识符的命名往往不被初学者重视，但是养成良好的命名习惯将使你的程序设计生涯获益匪浅。标识符的恰当应用意味着你的整个程序具有良好的可读性和逻辑性，因此在给每一个变量、每一个函数命名时一定要慎重，并遵循一定的规则。下列是关于标识符命名的一些技巧。

1. 名副其实

标识符虽然可由我们自己随意定义，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，做到“名副其实”。例如一个变量单纯起名为d，就不如begindate、enddate之类的更容易让人理解。下面的程序，比例3-1就更容易理解。

例3-3 较直观的标识符。

```
4    char upperchar,lowerchar;
5    printf("请输入一个大写字母:\n");
6    scanf("%c",&upperchar);
```

```
7     lowerchar = upperchar + 32;
8     printf("该大写字母对应的小写字母是: %c\n", lowerchar);
```

2. 方便阅读

容易阅读的标识符就更容易记忆。学过英语的人都知道,发音越标准越能够帮助我们记忆单词。方便阅读的变量使程序更加通顺,也能够帮助记忆变量。并且程序设计过程中,往往需要和其他人进行交流和讨论,容易发音阅读的标识符也有助于程序设计人员之间的交流。

3. 长度

虽然一般来讲,我们给变量或函数起名字不太可能超过系统规定的长度,但还是要注意不要使用过长的变量名字,例如,标准 C 不限制标识符的长度,但它受各种版本的 C 语言编译系统限制,同时也受到具体机器的限制,在某版本 C 中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。

通常可以使用一些容易看懂的缩写加上下画线来命名,这样有利于缩减标识符的长度,同时又能够名副其实。例如,beginTimeOfProjectLesson 可以缩写为 b_timePL,这样既能区分开变量,又能够给人以提示。

4. 有意义

标识符应该有一定的意义和代表性。最常见的,在 C 语言中,通常用 ch 来表示一个字符型变量,用 i 来表示一个整数类型的变量。这是因为 char 和 int 两个关键字分别代表了字符型和整数型,这样一下子就能够看出区别。如果一个程序很短,并且意思比较清晰明确,可以采用这样简单的命名方式,但是一旦程序较长,那么尽量使用能够区分开来,并且容易进行搜索的名字。例如,在一段程序中搜索 upperchar 和 ch 结果会完全不同。

关于表示符命名还有很多的地方需要注意,在以后程序设计中,必须注意命名,并增加积累,养成良好的习惯。

3.1.3　数据类型

计算机程序的运行过程,就是对数据进行加工处理的过程。这里的数据是广义的,可以是数字、文字、声音、图片以及视频等不同种类的数据。这些不同种类的数据在计算机内部的存储方式以及处理方法是不同的,但同一种类的数据在计算机内部的表示以及处理的方法则是相同的。

如果程序设计语言中能够区分各种数据的种类,那么在编程时就可以根据需要,选择合适的类型,大大地方便了程序的编写。这就好比交通工具分为火车、汽车、飞机、轮船等不同种类一样,它们各有各的特点。当人们要出行时,就可以根据实际路况选择适合的交通工具。

不同的语言数据类型有所不同,常用的数据类型可分为两大类:基本数据类型和导出数据类型。基本数据类型包括整型、实型、字符型、字符串型、布尔型等,导出数据类型是由基本数据类型构造出来的数据类型,包括枚举、数组、结构体、指针、类等。如图 3-4 所示。

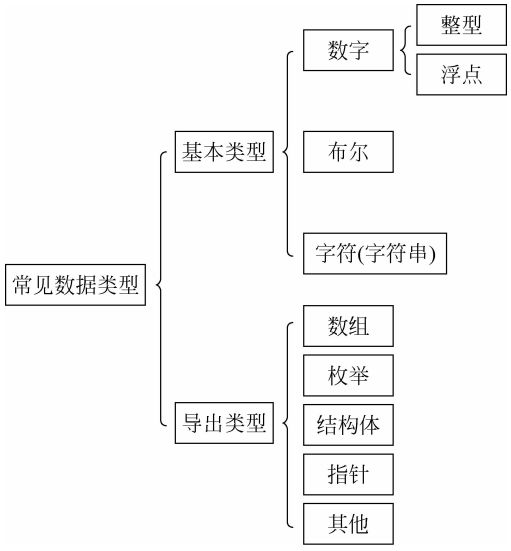

图 3-4　常见的数据类型

3.2　常量与变量

在表示符中，最常见的就是常量和变量，它们是构成一个程序的基础。下面详细讨论一下程序设计中关于常量和变量的内容。

3.2.1　常量

常量是指在程序的运行过程中其值不发生改变的量，数值常量就如同数学中的常数。根据常量的数据类型的不同，常量可分为整型常量、实型常量、字符型常量、字符串常量以及符号常量 5 种。

1. 整型常量

不带小数点的数值称为整型常量。例如，5、12106、0、－333 等都是常见的整型常量。整型常量按其采取的进制的不同，可分为八进制、十进制和十六进制常量。

(1) 八进制常量

以数字 0 开头，例如 012 和 034，分别对应十进制 10 和 28。

(2) 十进制常量

没有任何的前导符，例如 10、－1 及例 3-1 中的 32。

(3) 十六进制常量

以 0x 开头，例如 0x12 和 0x45，分别对应十进制的 18 和 69。

2. 实型常量

以小数形式或指数形式表示的数值称为实型常量，也称为浮点型常量。

(1) 小数形式

由数字和小数点组成，例如 12.34、0.1、－6.54 等。

(2) 指数形式

由尾数、字母 e 或 E 和指数组成，例如 1.23e3(代表 1.23×10^{3})、－45.21E－5(代表 -45.12×10^{-5})等。书写时应注意，e 或 E 的前面必须有数字，并且 e 或 E 的后面的数必须是整数，例如 e3、1E2.5 等都不是正确的指数形式。

3. 字符型常量

一共有两种类型的字符型常量：一种是单个字符；另一种是转义字符。字符型常量在使用的时候一般用单引号引起来，如果用双引号，则表示是一个字符串常量，注意与后面字符串常量比较。

(1) 单个字符

单一字符，用一对单引号('')包围，例如'a'、'0'及'￥'等。注意字符型常量必须是单个字符，并且字符与 ASCII 表中字符对应。对于 C 语言来讲，字符是以 ASCII 码值的形式存放在内存中的，也就是说图 3-3 中内存中存放的并不是'A'，而是 65。

所以，'A'和'a'是完全不同的两个字符，不存在'Aa'或'01'这样的字符。'0'、'1'是两个字符。

(2) 转义字符常量

除了 ASCII 表中的一些字符外，在 C 语言中还存在一些转义字符常量，以反斜杠(\)后跟一个字符或一个数值(八进制或十六进制)表示。常见的转义字符如表 3-1 所示，其作用主要是用来进行格式输入或者格式输出。例如，在 printf 函数中用'\n'来表示换行，用'\b'表示回退等。

表 3-1　C 语言常用的转义字符及其含义

转义字符	含　义	转义字符	含　义	转义字符	含　义
\a	响铃	\b	回退，向后退一格	\f	换页
\n	换行，将光标移到下行行首	\r	回车，将光标移到本行行首	\t	水平制表
\v	垂直制表	\\	反斜杠	\'	单引号
\"	输出双引号	\ddd	三位八进制	\xhh	二位十六进制

4. 字符串常量

字符串常量是一般程序设计语言中最常见的，用一对双引号("")引起来的一个或多个字符，例如"hello"、"a"等。

思考：'A'和"A"是否相同？

5. 符号常量

符号常量可以理解为对常量命名，即用符号代替常量。在 C 语言中，符号常量需要"先定义，后使用"，习惯上符号常量用大写字母表示。

定义符号常量的格式为：

```
#define 符号常量
```

例如，如果程序中要分别求出球体、圆柱体和圆锥体的表面积和体积，则需要多次用到圆周率 π 的值，这种情况下，就可以将 π 的值定义成一个符号常量，当需要用 π 时就用符号

常量代替它，即：

```
#define PI 3.1416                    //注意此行末尾没有分号
```

当编译系统对程序编译前，即预编译时，预处理器将程序中的所有符号常量 PI 置换成 3.1416。

例 3-4 符号常量的运用举例。

程序代码：

```
#include <stdio.h>
#define PRICE 30
main( )
{
  int num,total;
  num = 10;
  total = num * PRICE;
  printf("total = %d",total);
  system("pause");
}
```

使用符号常量有什么好处呢？

第一，增强程序的可读性。

在定义符号常量时，尽量遵循“见名知意”的原则，即从符号常量的名字就可以清楚地看出该常量的含义。

例如，上述内容中定义的符号常量 PI，读程序的人很容易由符号常量名 PI 知道这个常量的含义就是圆周率。当然要想达到这样的效果，前提是符号常量名必须和其实际的含义相符合。如果用“#define PRICE 3.14159”来定义一个符号常量 PRICE 代替 3.14159，就失去符号常量的存在意义了；而如果改为“#define PRICE_APPLE 3.8”，则从符号常量的名字就可以清楚地看出其代表苹果的价格，达到“见名知意”的效果。

第二，增强程序的可维护性，避免多处修改。

如果在程序中直接使用常量，当一个常量的值需要修改时，那么很显然程序的修改工作量就很大，尤其是当这个常量出现次数很多的时候。但是，如果用符号常量来实现，那么只需要改动一个地方，就是#define 命令这个地方，而程序其他部分完全不需改动，因为在进行预编译时新的常量值会被正确地替换进去。

例如，如果之前定义的符号常量 PI 代表圆周率 3.1416，而后期需要将圆周率的值修改为 3.14159，则只需将“#define PI 3.1416”改为“#define PI 3.14159”即可，达到“一改全改”的效果。

3.2.2 变量

任何一种编程语言都离不开变量，变量是计算机编程中的一个重要概念，程序处理的数据大部分是以变量的形式存在的，尤其是数据处理型程序，变量的使用非常频繁，离开变量程序甚至无法编写。那么，变量的概念是什么呢？怎样来理解变量呢？

3.2.2.1 变量的内涵

其实到目前为止，我们已经多次接触过变量，例如，例 2-1 中的 n 和 i 以及例 3-1 中的 ch。

到底什么是变量呢？顾名思义，变量就是在程序的运行期间值可以改变的量。例如，如果设计一个程序来实时反映罐子里巧克力豆的数目，因为巧克力豆是要被吃掉的，所以罐里巧克力豆的数目可能会随着时间的推移而改变，这种情况下就可以使用一个变量来跟踪不同时间的巧克力豆数目。

变量由变量名和变量值两部分组成。变量名代表分配给该变量的存储单元，变量的值就是存放在这个存储单元中的数据。例如，若变量名为 a，变量的值为 5，那么变量名、变量的值以及存储单元用图表示的结果如图 3-5 所示。

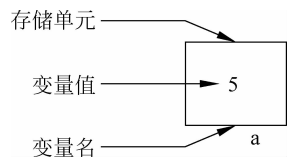

图 3-5　变量名、变量值及存储单元示意图

从图中可以看出，变量名实际上是以一个名字代表的一个存储地址，从变量中取值，实际上是通过变量名找到相应的内存地址，从该内存地址对应的存储单元中读取数据。

为了更形象，我们可以这样理解变量：将变量看做教学楼的房间，即存储单元；每个房间都有其唯一的名字，例如机房、教室、实验室等，这就相当于变量的名字；每个房间里都可以放置物品，这就相当于变量的值。

除此之外，大家再思考一个问题：每个房间放置的物品是相同的吗？显然不是，机房里放的是计算机，教室里放的是书桌坐椅，而实验室里放的是实验用品。变量也是一样，不同的变量中可以存放不同类型的值。例如，例 3-1 中的 ch 中存放的是字符，而例 2-1 中的 n 和 i 中存放的都是整数，这就是变量的数据类型。

每个变量都有其数据类型，在变量定义时指定。数据类型限制了变量中只能存放指定类型的数据，例如，如果在定义时指明了变量 str 是字符串类型，那么 str 中只能放字符串类型的数据，如果放入其他类型的数据，在程序编译时就会提示出错。正如“实验室里只能放实验用品，而不能放计算机或者书桌坐椅”一样。

3.2.2.2　变量的定义

变量的定义就是给要使用的存储单元起名的过程，就像一个教室，在没有指定房间名无法找到一样，所有的变量必须“先定义，后使用”。在起名的过程中，必须要指明变量的数据类型。

在 C 语言中，变量的定义格式为：

```
数据类型说明符　变量名 1,变量名 2,...;
```

如果多个变量属于同一种数据类型，则可以同时定义多个变量，每个变量名之间用逗号“,”间隔。

在定义变量时指定其数据类型，限制了变量的值只能是指定的类型的数据，这是数据类型的作用之一。除此之外，数据类型还决定了变量所分配的存储单元的字节数。编译系统在对程序进行编译时，根据其数据类型分配给变量相应的字节数，字节数又决定了其可存储的数据的范围。接下来介绍一下常见的整型、实型和字符型变量。

1. 整型变量

程序中最常见的变量类型即整型变量，即用内存中的一个存储空间来存储一个整数。需要注意的是，由于内存空间分配的存储单元大小有限，因此所能存储的整数大小是有限的。理论上讲，无论功能多么强大的计算机，也不能把一个无限大的整数完整地表示出来。

对于C语言来讲，一般操作系统给基本整型(int)变量分配2个字节的存储空间，因此，C语言中int能表示数的范围为－32768～32767，即-2^{15}～$(2^{15}-1)$。

表3-2列出了C语言中的整型类型说明符和表示数字的范围。

表3-2　整型变量

类型说明符	数的范围		字节数
int	－32768～32767	即-2^{15}～$(2^{15}-1)$	2
unsigned int	0～65535	即0～$(2^{16}-1)$	2
short int	－32768～32767	即-2^{15}～$(2^{15}-1)$	2
unsigned short int	0～65535	即0～$(2^{16}-1)$	2
long int	－2147483648～2147483647	即-2^{31}～$(2^{31}-1)$	4
unsigned long	0～4294967295	即0～$(2^{32}-1)$	4

其中，int为基本整型，unsigned int为无符号整型，short int为短整型，unsigned short int为无符号短整型，long int为长整型，unsigned long为无符号长整型。具体使用方式如下：

```
int a,b,c; (a,b,c 为整型变量)
long x,y; (x,y 为长整型变量)
unsigned p,q; (p,q 为无符号整型变量)
```

在书写变量定义时，应注意以下几点。

- 允许在一个类型说明符后，定义多个相同类型的变量。各变量名之间用逗号间隔。类型说明符与变量名之间至少用一个空格间隔。
- 最后一个变量名之后必须以“；”号结尾。
- 变量定义必须放在变量使用之前。一般放在函数体的开头部分。

例3-5　整型变量的定义与使用。

程序代码：

```
main()
{
  int a,b,c,d;
  unsigned u;
  a = 12;b = -24;u = 10;
  c = a + u;d = b + u;
  printf("a + u = %d,b + u = %d\n",c,d);
  system("pause");
}
```

2. 实型变量

在用于处理数字类型的变量中，除了整数类型，还有实数类型，在C语言中也成为浮点型变量，根据所占据的存储单元的多少又分为单精度和双精度两种类型。

在C中单精度型占4个字节(32位)内存空间，其数值范围为3.4E－38～3.4E＋38，只能提供七位有效数字。双精度型占8个字节(64位)内存空间，其数值范围为1.7E－308～1.7E＋308，可提供16位有效数字。

C语言中的浮点类型说明符和表示数字的范围见表3-3。

表 3-3　C 语言中的浮点类型说明符和表示数字的范围

类型说明符	比特数(字节数)	有效数字	数的范围
float	32(4)	6～7	10^{-37}～10^{38}
double	64(8)	15～16	10^{-307}～10^{308}
long double	128(16)	18～19	10^{-4931}～10^{4932}

实型变量定义的格式和书写规则与整型相同。

例如：

```
float x,y; (x,y 为单精度实型量)
double a,b,c; (a,b,c 为双精度实型量)
```

需要注意的是计算机处理实型数据有舍入误差，由于实型变量是由有限的存储单元组成的，因此能提供的有效数字总是有限的。

例 3-6　实型数据的舍入误差。

程序代码：

```
main()
{
   float a,b;
   a = 123456.789e5;
   b = a + 20;
   printf(" % f\n",a);
   printf(" % f\n",b);
   system("pause");
}
```

程序运行结果如图 3-6 所示。

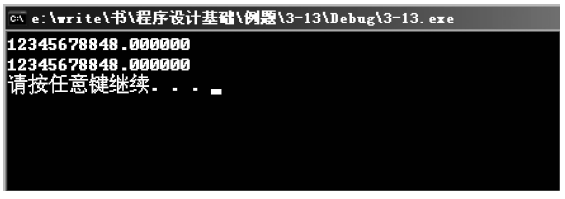

图 3-6　例 3-6 的运行结果

又例如，1.0/3 * 3 的结果并不等于 1。实际上，在计算机中，是不能直接存储分数的，需要将分数转化为小数来计算，这样就容易产生误差。

例 3-7　进一步理解浮点。

程序代码：

```
main( )
{
   float a;
   double b;
   a = 33333.33333;
   b = 33333.33333333333333;
   printf(" % f\n % f\n",a,b);
}
```