第3章

顺序结构程序设计

结构化程序可以由顺序、选择和循环三种基本结构组成。所谓顺序结构是指程序中的语句完全按照其排列次序执行。本章主要介绍顺序结构程序的编写以及一些常用语句。

3.1 C 语言的语句类型

C 语言中的语句可以分成以下 6 种类型。

(1) 说明语句,是用于定义或声明变量、数组、类型以及函数原型的语句。例如:

```
int a;
float x;
```

(2) 表达式语句,在表达式之后加一个分号构成。例如:

```
a = a + 1;
i + + ;
```

赋值语句是最重要的表达式语句。

(3) 函数调用语句,在函数调用之后加一个分号构成。例如:

```
printf("x = %d, y = %d\n",x,y);
scanf("%d%d",&x,&y);
```

(4) 空语句,即只由一个分号构成的语句。空语句不执行任何操作,通常用于一些特殊场合。

(5) 控制语句,是用于控制程序执行流程的语句。如 if-else 语句、while 语句等。

(6) 复合语句,是用一对大括号括起来的若干条语句。例如:

```
{t = a;a = b;b = t;}
```

从语法作用上来说,一条复合语句视为一条语句。

3.2 变量的赋值和初始化

3.2.1 赋值表达式

赋值表达式的一般形式为:

```
变量 = 表达式
```

其中的“＝”称为赋值运算符。

赋值表达式的功能是先求得赋值运算符右侧表达式的值，再将该值存入到左侧的变量中。

例如：

```
a = 3
a = a + 1
```

可以出现在赋值运算符左侧的量，称为左值(Lvalue)。变量是最常见的左值。

作为一种表达式，赋值表达式也有自己的值。一个赋值表达式的值，其实就等于赋值运算符左侧变量的值。

【例 3.1】 赋值表达式的值示例。

```
#include <stdio.h>
int main(void)
{
 int a,b;
 a = 3;
 printf("%d\n",b = a);                    /* 输出表达式 b = a 的值 */
 return 0;
}
```

程序的运行结果为：

```
3
```

赋值运算符右侧的表达式也可以是赋值表达式。

【例 3.2】 赋值运算符的结合性。

```
#include <stdio.h>
int main(void)
{
 int a,b,c;
 a = b = c = 6;
 printf("a = %d,b = %d,c = %d\n",a,b,c);
 return 0;
}
```

可见，赋值运算符具有右结合性。

3.2.2　变量的初始化

在 C 语言中，允许在定义变量的同时给变量赋值，称为变量的初始化。

【例 3.3】 变量的初始化示例。

```
#include <stdio.h>
int main(void)
{
 int a = 3,b = 6;
```

```
 printf("a = %d,b = %d\n",a,b);
 return 0;
}
```

【例 3.4】 变量的初始化错例。

```
#include <stdio.h>
int main(void)
{
 int a = b = c = 6;
 printf("a = %d,b = %d,c = %d\n",a,b,c);
 return 0;
}
```

该程序运行时,将会出现变量 b、c 未定义的错误。因为在这种格式中,C 语言编译系统会认为变量 b、c 未经定义而直接引用。

正确的程序如下所示:

```
#include <stdio.h>
int main(void)
{
 int a = 6,b = 6,c = 6;
 printf("a = %d,b = %d,c = %d\n",a,b,c);
 return 0;
}
```

3.3 数据的格式输入与格式输出

除了赋值语句之外,程序中用得最多的就是数据的输入和输出了。C 语言中数据的输入与输出均是由库函数实现的。常用的输入与输出函数包括 printf 函数、scanf 函数、putchar 函数和 getchar 函数等。

在程序中调用输入与输出库函数时,应在程序的开头添加以下文件包含命令:

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

3.3.1 格式输出函数(printf 函数)

printf 函数用于按照指定的格式向标准输出设备(通常是显示器)输出数据。

1. printf 函数的一般形式

```
printf(格式控制字符串,输出项表)
```

其中的输出项表是若干个要输出的数据项(可以是常量、变量或表达式),而格式控制字符串则用于规定输出项的输出格式。

【例 3.5】 printf 函数示例。

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

```
int main(void)
{
 int a = 100;
 float x = 123.456;
 printf("a = %d,x = %f\n",a,x);
 return 0;
}
```

格式控制字符串中的字符包括以下两部分：

(1) 格式说明，由“%”和格式说明字符组成，如%d、%f 等，用于规定与之对应的数据项的输出格式。

如上例中的%d 对应于变量 a，%f 对应于变量 x。

(2) 普通字符，是格式说明以外的字符，普通字符将原样输出。

如上例中的“a=”“,”“x=”“\n”等都是普通字符。

故上例的运行结果为：

```
a = 100,x = 123.456001
```

2. 常用格式说明字符

1) d 格式符

d 格式符用于输出有符号十进制整数，包括以下几种用法。

① %d

%d 按实际长度输出有符号十进制整数。

② %md

%md 中的 m 为正整数，用于指定输出位数。

【例 3.6】 指定整数的输出位数。

```
#include <stdio.h>
int main(void)
{
 int a = 123,b = 123456;
 printf("%d,%d\n",a,b);
 printf("%4d,%4d\n",a,b);
 return 0;
}
```

运行结果为：

```
123,123456
␣123,123456
```

可见，当实际位数超过指定位数时，将按实际位数输出，指定位数不起作用；从而避免造成有效数据的丢失。

③ %ld

%ld 用于输出 long int 型数据。

【例 3.7】 输出长整型数据。

```
#include <stdio.h>
int main(void)
{
 long a = 1234567890;
 printf("%ld\n",a);
 return 0;
}
```

④ %hd

%hd 用于输出 short int 型数据。

2) f 格式符

f 格式符用于以十进制小数形式输出实数,包括以下几种用法。

① %f

%f 的整数部分按实际长度输出,并固定输出 6 位小数。

② %m.nf

%m.nf 中的 m、n 均为正整数。m 用于指定输出实数的总位数,n 用于指定小数位数。

【例 3.8】 指定实数的输出位数和输出精度。

```
#include <stdio.h>
int main(void)
{
 float x = 123.456;
 printf("%f\n",x);
 printf("%.2f\n",x);
 printf("%9.2f\n",x);
 printf("%6.4f\n",x);
 return 0;
}
```

运行结果为:

```
123.456001
123.46
␣␣␣123.46
123.4560
```

可见,当实际位数超过指定的总位数时,将按实际位数输出,指定的总位数不起作用。从而避免造成整数部分的有效数据丢失。

③ %lf

%lf 用于输出 double 型数据。

【例 3.9】 double 型数据的输出。

```
#include <stdio.h>
int main(void)
{
 double x = 123.456;
 printf("%lf\n",x);
```

```
 printf("%f\n",x);
 return 0;
}
```

运行结果为：

```
123.456000
123.456000
```

可见，double 型数据既可以用%lf 格式符输出，也可以用%f 格式符输出。

更多的 printf 函数格式符请参考表 3.1，printf 函数中使用的附加格式说明符见表 3.2。

表 3.1 printf 函数中使用的格式说明符

输出类型	格式说明符	说　明
整型数据	%d(或%i)	以有符号十进制形式输出整型数
	%o	以无符号八进制形式输出整型数(不输出前导 0)
	%x(或%X)	以无符号十六进制形式输出整型数(不输出前导 0x)
	%u	以无符号十进制形式输出整型数
实型数据	%f	以小数形式输出单精度、双精度的实型数
	%e(或%E)	以指数形式输出单精度、双精度的实型数
字符型数据	%c	输出一个字符
	%s	输出一个字符串
其他	%%	输出字符%本身

表 3.2 printf 函数中使用的附加格式说明符

符　号	说　明
l	输出长整型数或双精度实型数
h	输出短整型数
m	指定数据输出的宽度(即域宽)
.n	对按%f 或%e 输出的实型数据，指定输出 n 位小数
+	使输出的数值数据无论正负都带符号输出
−	使数据在输出域内按左对齐方式输出

3. 字符串常量的输出

用 printf 函数输出字符串常量时，可以不使用格式控制字符，而直接输出字符串常量。

【例 3.10】 用 printf 函数输出字符串。

```
#include <stdio.h>
int main(void)
{
 printf("How are you?\n");
 printf("%s","How are you?\n");
 return 0;
}
```

运行结果为：

```
How are you?
How are you?
```

可见，输出字符串常量时，不显示作为定界符的双引号。

【例 3.11】 已知一年期存款的本金和年利率，若到期后将本息自动转存，编程序求出存满 n 年的本息之和是多少。

编程思路：

这里实际上就是按复利计算利息，也就是将本期的利息自动转入到下一期的本金中。因此，可以直接利用复利公式 $f=p*(1+r)^n$ 求解。

源程序：

```
#include <stdio.h>
#include <math.h>
int main(void)
{
 float p,r,f;
 int n;
 printf("请输入本金、利率和年数: \n");        /* 输出提示信息 */
 scanf("%f%f%d",&p,&r,&n);
 f = p * pow(1 + r,n);                        /* 存满 n 年的本息之和 */
 printf("本息之和 = %f\n",f);
 return 0;
}
```

3.3.2 格式输入函数(scanf 函数)

scanf 函数用于从标准输入设备(通常是键盘)输入数据，并存入到指定的变量中。

1. scanf 函数的一般形式

```
scanf(格式控制字符串,变量地址表)
```

例如：

```
scanf("%d%d",&a,&b);
```

其中的变量地址表是若干个用于存储数据的变量的地址。格式控制字符串用于规定变量的输入格式，其用法与 printf 函数中的格式控制字符串类似。

2. 常用格式说明字符

与 printf 函数中的格式说明类似，常用 scanf 函数格式符请参考表 3.3。

表 3.3 scanf 函数中使用的格式说明符

输入类型	格式说明符	说　明
整型数据	%d	输入十进制整型数
	%u	输入无符号的十进制整型数
	%o	输入八进制整型数
	%x	输入十六进制整型数

续表

输入类型	格式说明符	说　　明
实型数据	%f	输入小数形式的单精度实型数
	%e	输入指数形式的单精度实型数
字符型数据	%c	输入单个字符
	%s	输入一个字符串

scanf 函数中使用的附加格式说明符如表 3.4 所示。

表 3.4　scanf 函数中使用的附加格式说明符

符　　号	说　　明
l	输入长整型数或双精度实型数
h	输入短整型数

使用 scanf 函数时，要注意以下几个问题。

(1) 格式控制字符串中的普通字符，必须原样输入。

【例 3.12】 不使用普通字符的 scanf 函数。

```
#include <stdio.h>
int main(void)
{
 int x,y;
 scanf("%d%d",&x, &y);
 printf("x= %d,y= %d\n",x,y);
 return 0;
}
```

程序运行时，应输入

```
3␣6
```

【例 3.13】 使用普通字符的 scanf 函数。

```
#include <stdio.h>
int main(void)
{
 int x,y;
 scanf("%d,%d",&x, &y);
 printf("x= %d,y= %d\n",x,y);
 return 0;
}
```

程序运行时，应输入

```
3,6
```

【例 3.14】 使用普通字符的 scanf 函数。

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

```
int main(void)
{
 int x,y;
 scanf("x = %d,y = %d",&x, &y);
 printf("x = %d,y = %d\n",x,y);
 return 0;
}
```

程序运行时,应输入

```
x = 3,y = 6
```

尤其要注意,在 scanf 函数的格式控制字符串中,不应出现“\n”。

【例 3.15】 scanf 函数错例。

```
#include <stdio.h>
int main(void)
{
 int x,y;
 scanf("%d%d\n",&x, &y);
 printf("x = %d,y = %d\n",x,y);
 return 0;
}
```

其中的“scanf ("%d%d\n",&x, &y);”语句虽然算不上有语法错误,但是该语句运行时不能正常退出。

【例 3.16】 已知一元二次方程 $ax^2+bx+c=0$ 的系数 a、b、c 的值,设 $b^2-4ac\geqslant 0$ 且 $a\neq 0$,编程序求该方程的根。

编程思路:

不能通过直接将方程式写在程序中来求解方程。可以利用一元二次方程的求根公式,分别求解该方程的两个根。

源程序:

```
#include <stdio.h>
#include <math.h>
int main(void)
{
 float a,b,c,p,q,x1,x2;
 printf("请输入三个系数的值: \n");
 scanf("%f%f%f",&a,&b,&c);
 p = -b/(2*a);                          /*尽量避免重复性计算*/
 q = sqrt(b*b-4*a*c)/(2*a);             /*不要丢失乘号和括号*/
 x1 = p+q;
 x2 = p-q;
 printf("x1 = %f,x2 = %f\n",x1,x2);
 return 0;
}
```

3.4 拓展：赋值运算中的类型转换

在C语言中，若两种类型的数据之间可以直接相互赋值，则称之为赋值兼容。例如，整型、实型及字符型之间是赋值兼容的。当赋值运算符两侧的数据类型不一致时，将会依据左侧变量的类型，自动地对右侧表达式的值进行类型转换。

以下是三种最常用的赋值转换。

3.4.1 实型数据赋给整型(或字符型)变量

当实型数据赋给整型(或字符型)变量时，将会对实型数据进行截断取整。

【例3.17】 实型数据赋给整型变量。

```
#include <stdio.h>
int main(void)
{
 int a;
 a = 3.5678;
 printf("a = %d\n",a);
 return 0;
}
```

程序运行结果：

```
a = 3
```

3.4.2 整型(或字符型)数据赋给实型变量

当整型(或字符型)数据赋给实型变量时，将会把整型(或字符型)数据转换为实数格式并存入到实型变量中。

【例3.18】 整型数据赋给实型变量。

```
#include <stdio.h>
int main(void)
{
 float x;
 x = 99;
 printf("x = %f\n",x);
 return 0;
}
```

程序运行结果：

```
x = 99.000000
```

3.4.3 整型数据赋给类型不同的等长整型变量

当整型数据赋给类型不同但内存位数相同的整型变量时，将会按该数据的内部形式原

样传送。

【例 3.19】 整型数据赋给不同类型的等长整型变量。

```
#include <stdio.h>
int main(void)
{
 unsigned short a;
 short int b;
 b = -1;
 a = b;
 printf("a = %hu\n",a);                    /* %hu 用于输出无符号短整型数据 */
 return 0;
}
```

程序运行结果：

```
a = 65535
```

为什么会是这个结果呢？这是因为变量 b 为有符号短整数，故其内部形式为 1111 1111 1111 1111。将变量 b 的值赋给变量 a 之后，变量 a 的内部形式也是 1111 1111 1111 1111。但因为变量 a 为无符号短整数，故其十进制真值为 65 535。

3.5 项目式案例

【例 3.20】 已知地球的赤道半径为 6377.830km，并已知位于东半球赤道上两点的经度值(单位为(°))，编程序计算这两点之间的球面距离。

编程思路：

在赤道上两点之间的球面距离，就是这两点之间劣弧的长度。根据弧长公式，首先求出两点之间圆心角的大小(单位为 rad)，然后乘上赤道半径即可。

源程序：

```
#include <stdio.h>
#define PI 3.14159
int main(void)
{
 float r,a,b,t,arc;
 r = 6377.830;
 printf("请输入东半球两点的经度值(单位为度): \n");
 scanf("%f",&a);
 scanf("%f",&b);
 t = a - b;
 t = t/180 * PI;                          /* 单位转换 */
 arc = t * r;
 printf("两点之间的球面距离 = %f 千米\n",arc);
 return 0;
}
```

在上面的程序中，若第一个经度值小于第二个经度值，则经度差为负值，从而导致最终

的球面距离为负值。不过，可以利用绝对值函数解决这个问题。

改进版源程序：

```
#include <stdio.h>
#include <math.h>
#define PI 3.14159
int main(void)
{
 float r,a,b,t,arc;
 r = 6377.830;
 printf("请输入东半球两点的经度值(单位为度): \n");
 scanf("%f%f",&a,&b);
 t = fabs(a - b);
 t = t/180 * PI;                                  /*单位转换*/
 arc = t * r;
 printf("两点之间的球面距离 = %f千米\n",arc);
 return 0;
}
```