第 3 章　C 语言程序结构及相关语句

从程序流程的角度来看，程序可以分为 3 种基本结构：顺序结构、分支结构、循环结构。这 3 种基本结构可以组成所有各种复杂程序。C 语言提供了多种语句来实现这些程序结构。本章将介绍这些基本语句及其应用，为后面各章的学习打下基础。

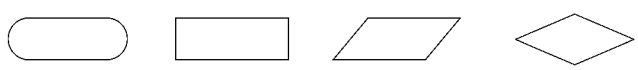
3.1　相关知识

3.1.1　算法描述方法

算法一般可以用以下两种方法来进行描述。

(1) 伪代码。采用近似高级语言但又不受语法约束的语言描述。

(2) 流程图。传统的流程图常用的符号如图 3.1 所示，由这些框和流程线组成的流程图来表示算法，形象直观，简单方便，但在描述复杂算法时不易阅读。



(a) 开始或结束框　(b) 处理框　(c) 输入输出框　(d) 判断框

图 3.1　流程图常用符号

3.1.2　结构化程序

结构化程序设计方法是程序设计的先进方法，结构化程序设计是一种使用顺序、选择和重复共 3 种基本控制结构，并且使用这 3 种基本结构足以表达出各种其他形式的结构的程序设计方法。

(1) 顺序结构。在执行时按照先后顺序逐条进行，没有分支，没有循环。如后面介绍的赋值语句、输入输出语句等都可以构成顺序结构。顺序结构可用图 3.2 所示的流程图来表示。

(2) 选择结构。根据不同的条件去执行不同分支中的语句。如后面章节中介绍的 if 语句、switch 语句等都可以构成选择结构。选择结构可用图 3.3 所示的流程图来表示。

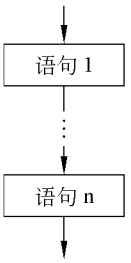


图 3.2 顺序结构流程图

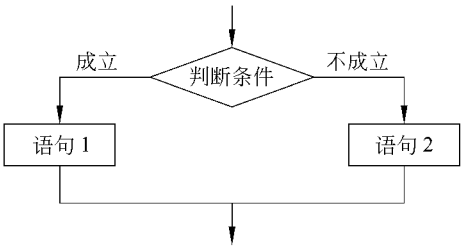
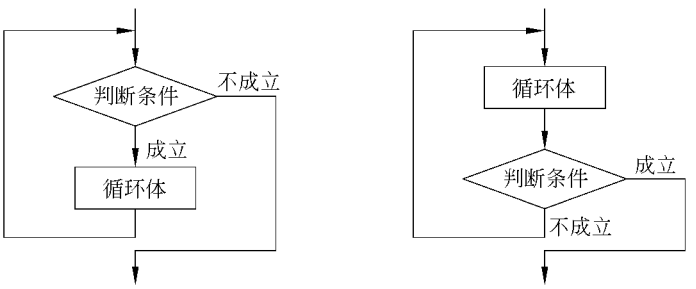


图 3.3 选择结构流程图

(3) 循环结构。根据各自的条件,使同一组语句重复执行多次或一次也不执行。循环结构包括当型循环(如图 3.4(a)所示)和直到型循环(如图 3.4(b)所示)。当型循环的特点是,当指定的条件满足时,就执行循环体,否则就不执行。直到型循环的特点是,执行循环体直到指定的条件满足,就不再执行循环。



(a) 当型循环　　(b) 直到型循环

图 3.4 选择结构流程图

*3.1.3 模块化结构

计算机在处理复杂任务时,常常需要把大任务分解为若干个子任务,每个子任务又分成很多个小子任务,每个小子任务只完成一项简单的功能。在程序设计时,用一个个模块来实现这些功能。这样的程序设计方法称为“模块化”,由一个个功能模块构成的程序结构就称为“模块化结构”。模块化结构可以大大提高程序编制的效率。

C 语言是一种结构化程序设计语言。它直接提供了 3 种基本结构的语句;提供了定义“函数”的功能,用函数实现模块化结构。

3.2 顺序语句

3.2.1 C 程序的语句

C 程序的执行部分由语句组成的。程序的功能也由执行语句来实现。C 语句可分 5 类:表达式语句、函数调用语句、控制语句、复合语句和空语句。

1. 表达式语句

表达式语句由表达式加上分号(;)组成。一般形式为：

```
表达式;
```

执行表达式语句就是计算表达式的值。例如：

```
c=a+b;
```

是赋值语句。

```
a+ b;
```

是加法运算语句，计算结果不能保留，无实际意义。

```
i++;
```

是自增1语句，i值增1。

2. 函数调用语句

由函数名、实际参数加上分号(;)组成。一般形式为：

```
函数名(实际参数表);
```

执行函数语句是调用函数体并将实际参数赋予函数定义中的形式参数，然后执行被调函数体中的语句，求取函数值。

3. 控制语句

控制语句用于控制程序的流程，它们由特定的语句定义符组成，用以实现程序的各种结构方式，在后面章节将进行详细介绍。

4. 复合语句

将多个语句用一对大括号({})括起来组成的一个语句称为复合语句。在程序中应把复合语句看成是单条语句，而不是多条语句，例如：

```
{
    k=i+j;
    a=b-c;
    printf("%d,%d", k, a);
}
```

是一条复合语句。复合语句内的各条语句都以分号(;)结尾，在大括号(})外不用加分号。

5. 空语句

只有分号(;)组成的语句称为空语句。空语句什么也不执行。在程序中空语句可用作空循环体。例如 while(getchar()!='\n'); 本语句的功能是，只要从键盘输入的字符(用“getchar()”实现)不是回车符则重新输入。这里的循环体为空语句。

3.2.2　数据输出语句

数据输出语句用于向标准输出设备显示器输出数据。在C语言中，所有数据输入输

出都由库函数完成。因此都是函数语句。本节介绍 printf 函数和 putchar 函数。

1. 格式化输出函数 printf()

printf()函数称为格式输出函数,其中函数名最后一个字母 f 就为“格式”(format)的意思。printf()函数的功能是按用户指定的格式,将指定的数据显示到显示器屏幕上。

1) printf()函数调用的一般形式

printf()函数为一个标准库函数,函数原型在头文件 stdio.h 中。但作为一个特例,不要求在使用 printf ()函数之前必须包含 stdio.h 文件。printf 函数调用的一般形式为:

```
printf("格式控制字符串",输出表)
```

其中,格式控制字符串用于指定输出格式。格式控制串由格式字符串和非格式字符串两种组成。格式字符串是以%开头的字符串,在%后面跟有各种格式字符,以说明输出数据的类型、长度、小数位数等。如%d 表示按十进制整型输出,%ld 表示按十进制长整型输出,%c 表示按字符型输出等。后面将专门给予讨论。

非格式字符串在输出时按原样显示,在显示中起提示作用。输出表中给出了各个输出项,要求格式字符串和各输出项在数量和类型上应一一对应。

例 3.1 printf()函数示例。

```
/*文件路径名:e3_1\main.c*/
#include <stdio.h>                    /*包含库函数 printf()所需要的信息*/
#include <stdlib.h>                   /*包含库函数 system()所需要的信息*/

int main(void)                        /*主函数 main()*/
{
    int a=65, b=66;                   /*定义变量*/
    printf("%d %d\n", a, b);          /*格式化输出 a, b*/
    printf("%d,%d\n", a, b);          /*格式化输出 a, b*/
    printf("%c,%c\n", a, b);          /*格式化输出 a, b*/
    printf("a=%d,b=%d\n", a, b);      /*格式化输出 a, b*/

    system("PAUSE");                  /*调用库函数 system(),输出系统提示信息*/
    return 0;                         /*返回值 0, 返回操作系统*/
}
```

程序运行时屏幕输出如下:

```
65 66
65,66
A,B
a=65,b=66
请按任意键继续...
```

本例程序中 4 次输出了 a,b 的值,但由于格式控制串不同,输出的结果也不相同。第 1 个 printf()函数输出语句的格式控制串中,两格式串%d 之间加了一个空格(非格式

字符),输出的 a,b 值之间有一个空格。第 2 个 printf()函数输出语句的格式控制串中加入的是非格式字符逗号,因此输出的 a,b 值之间加了一个逗号。第 3 个 printf()函数输出语句的格式串要求按字符型输出 a,b 值。第 4 个 printf()函数输出语句为了提示输出结果又增加了非格式字符串。

2) 格式字符串

格式字符串的一般形式为:

[标志][输出最小宽度][.精度][长度]类型

其中方括号([])中的项为可选项。各项的意义介绍如下。

① 类型。类型字符用于表示输出数据的类型,其格式符和意义如表 3.1 所示。

表 3.1　printf()函数格式中的类型字符

类型字符	意　义
d, i	以十进制形式输出带符号整数(正数不输出符号)
o	以八进制形式输出无符号整数(不输出前缀 O)
x	以十六进制形式输出无符号整数(不输出前缀 OX)
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e	以指数形式输出单、双精度实数
g	以%f 和%e 中较短的输出宽度输出单、双精度实数
c	输出单个字符
s	输出字符串

② 标志。标志字符为－、＋、＃ 3 种,其意义如表 3.2 所示。

表 3.2　printf()函数格式中的标志字符

标志字符	意　义
－	结果左对齐,右边填空格
＋	输出符号(正号或负号)空格,输出值为正时冠以空格,为负时冠以负号
＃	对 c,s,d,u 类无影响;对 o 类,在输出时加前缀 o;对 x 类,在输出时加前缀 ox;对 e,g,f 类当结果有小数时才给出小数点

③ 输出最小宽度。用十进制整数来表示输出的最少位数,如果实际位数多于定义的宽度,则按实际位数输出,如果实际位数少于定义的宽度则补以空格或 0。

④ 精度。精度格式符以“.”开头,后跟十进制整数。具体的意义是,如果输出数字,则表示小数的位数;如果输出的是字符,则表示输出字符的个数;若实际位数大于所定义的精度数,则截去超过的部分。

⑤ 长度。长度格式符为 h,l 两种,h 表示按短整型量输出,l 表示按长整型量输出。

例 3.2 printf()函数格式字符串使用示例。

```
/*文件路径名:e3_2\main.c*/
#include <stdio.h>                          /*包含库函数 printf()所需要的信息*/
#include <stdlib.h>                         /*包含库函数 system()所需要的信息*/

int main(void)                              /*主函数 main()*/
{
    int a=168;                              /*定义整型变量*/
    float b=1243.1698;                      /*定义单精度实型变量*/
    double c=2421985.50168;                 /*定义双精度实型变量*/
    char d='a';                             /*定义字符型变量*/

    printf("a=%d,%5d,%o,%x\n", a, a, a, a);        /*输出 a*/
    printf("b=%f,%lf,%5.4lf,%e\n", b, b, b, b);    /*输出 b*/
    printf("c=%lf,%f,%8.4lf\n", c, c, c);          /*输出 c*/
    printf("d=%c,%8c\n",d,d);                      /*输出 d*/

    system("PAUSE");                    /*调用库函数 system(),输出系统提示信息*/
    return 0;                           /*返回值 0,返回操作系统*/
}
```

程序运行时屏幕输出如下：

```
a=168,  168,250,a8
b=1243.169800,1243.169800,1243.1698,1.243170e+003
c=2421985.501680,2421985.501680,2421985.5017
d=a,       a
请按任意键继续...
```

本例第 1 个 printf()函数以 4 种格式输出整型变量 a 的值，其中%5d 要求输出宽度为 5，而 a 值为 168 只有三位故补两个空格。第 2 个 printf()函数以 4 种格式输出实型量 b 的值。其中%f 和%lf 格式的输出相同，说明 l 符对 f 类型无影响。%5.4lf 指定输出宽度为 5，精度为 4，由于实际长度超过 5 故应该按实际位数输出，小数位数超过 4 位部分被截去。第 3 个 printf()函数输出双精度实数，%8.4lf 由于指定精度为 4 位故截去了超过 4 位的部分。第 4 个 printf()函数输出字符量 d，其中%8c 指定输出宽度为 8 故在输出字符'a'之前补加 7 个空格。

注意：*不同的编译系统输出表的求值顺序不一定相同，可以从左到右，也可从右到左。Visual C++ 6.0、Dev-C++ 和 MinGW Developer Studio 是按从右到左进行的。*

例 3.3 printf()函数输出表的求值顺序示例。

```
/*文件路径名:e3_3\main.c*/
#include <stdio.h>                          /*包含库函数 printf()所需要的信息*/
#include <stdlib.h>                         /*包含库函数 system()所需要的信息*/
```

```
int main(void)                          /* 主函数 main() */
{
    int i=6;                            /* 定义变量 */
    printf("%d,%d,%d\n",++i,++i,++i);/* 输出关于 i 的表达式之值 */

    system("PAUSE");                    /* 调用库函数 system(),输出系统提示信息 */
    return 0;                           /* 返回值 0, 返回操作系统 */
}
```

在 Visual C++ 6.0、Dev-C++ 和 MinGW Developer Studio 中，程序运行时屏幕输出如下：

```
9,8,7
请按任意键继续...
```

2. 字符输出函数 putchar()

putchar()函数用于输出字符，功能是在显示器上显示单个字符。函数原型在头文件 stdio.h 中，一般形式为：

```
putchar(字符变量)
```

例如：

putchar('a')；用于输出小写字母 a。

putchar(x)；用于输出字符变量 x 的值。

putchar('\n')；用于换行，对控制字符则执行控制功能，不在屏幕上显示。

例 3.4　putchar()函数使用示例。

```
/* 文件路径名:e3_4\main.c */
#include <stdio.h>                      /* 标准输入输出头文件 */
#include <stdlib.h>                     /* 包含库函数 system()所需要的信息 */

int main(void)                          /* 主函数 main() */
{
    char a='A', b='o', c='m';           /* 定义字符型变量 */
    putchar(a); putchar(b); putchar(b); putchar('\t');
                                                /* 输出字符, '\t'为制表符 */
    putchar(a); putchar(b); putchar('\n');      /* 输出字符, '\n'为换行符 */
    putchar(b); putchar(c); putchar('\n');      /* 输出字符, '\n'为换行符 */

    system("PAUSE");                    /* 调用库函数 system(),输出系统提示信息 */
    return 0;                           /* 返回值 0, 返回操作系统 */
}
```

程序运行时屏幕输出如下：

```
Aoo     Ao
```

```
om
请按任意键继续...
```

3.2.3 数据输入语句

C语言的数据输入是由函数语句完成。本节介绍从标准输入设备键盘上输入数据的函数scanf()和getchar()。

1. 格式化输入函数scanf()

scanf()函数称为格式输入函数,其功能是按用户指定的格式从键盘上把数据输入到指定的变量之中。

1) scanf()函数的一般形式

scanf()函数是一个标准库函数,它的函数原型在头文件stdio.h中,与printf()函数相同,C语言也允许在使用scanf()函数之前不必包含stdio.h文件。scanf()函数的一般形式为:

```
scanf("格式控制字符串",地址表);
```

其中,格式控制字符串的作用与printf()函数相同,但不显示非格式字符串,也就是不能显示提示字符串。地址表中给出各变量的地址。地址由地址运算符(&)后跟变量名组成的。例如,&x表示变量x的地址。此地址就是编译系统在内存中给x变量分配的地址。应该把变量的值和变量的地址这两个不同的概念区别开来。变量的地址是C编译系统分配的,用户不必关心具体的地址是多少。在赋值表达式中给变量赋值,如:x=168在赋值号左边是变量名,不能写地址,而scanf()函数要求写变量的地址,如&x。这两者在形式上是不同的。&是一个取地址运算符,&x是一个表达式,其功能是求变量的地址。

例3.5 scanf()函数使用示例。

```
/*文件路径名:e3_5\main.c*/
#include <stdio.h>                  /*标准输入输出头文件*/
#include <stdlib.h>                 /*包含库函数system()所需要的信息*/

int main(void)                      /*主函数main()*/
{
    int a, b;                       /*定义变量*/
    printf("输入a,b:");             /*输入提示*/
    scanf("%d%d", &a, &b);          /*输入a, b*/
    printf("a=%d,b=%d\n", a, b);    /*输出a, b*/

    system("PAUSE");                /*调用库函数system(),输出系统提示信息*/
    return 0;                       /*返回值0,返回操作系统*/
}
```

在本例中,由于scanf()函数本身不能显示提示,所以先用printf()语句在屏幕显示

输入提示，请用户输入 a,b 的值。执行 scanf()语句，等待用户输入。用户输入 6、8 后按下回车键，然后执行 printf()函数。在 scanf()函数的格式串中由于没有非格式字符在%d%d 之间作输入时的间隔，因此在输入时要用一个以上的空格、回车键或 Tab 符作为每两个输入数之间的间隔。

如：

```
6 8
```

或

```
6
8
```

程序运行时屏幕输出参考如下：

```
输入 a,b:6 8
a=6,b=8
请按任意键继续...
```

2）格式字符串

格式字符串的一般形式为：

```
%[*][输入数据宽度][长度]类型
```

其中有[]的项为任选项。各项的意义如下。

① 类型。表示输入数据的类型，其格式符和意义如表 3.3 所示。

表 3.3　scanf()函数格式中的类型字符

类型字符	意　义	类型字符	意　义
d, i	输入十进制整数	e,f	输入实型数(用小数形式或指数形式)
o	输入八进制整数	c	输入单个字符
x	输入十六进制整数	s	输入字符串
u	输入无符号十进制整数		

② 星号(*)。表示该输入项读入后不赋予相应的变量，即跳过该输入值。如 scanf("%d%*d%d",&a,&b);当输入为：6 8 9 时，把 6 赋予 a,8 被跳过,9 赋予 b。

③ 宽度。用十进制整数指定输入的宽度(即字符数)。例如：

```
scanf("%5d",&a);
```

输入：

```
1234567890
```

只把 12345 赋予变量 a,其余部分被截去。又如：

```
scanf("%4d%4d",&a,&b);
```

输入：

```
1234567890
```

将把 1234 赋予 a，而把 5678 赋予 b。

④ 长度。长度格式符为 l 和 h，l 用于表示输入长整型数据(如%ld) 和双精度浮点数(如%lf)。h 表示输入短整型数据。

使用 scanf()函数应注意以下几点：

① scanf()函数中没有精度控制，如：scanf("%5.2f",&a)；是非法的。用户不能企图用此语句输入小数为 2 位的实数。

② scanf()函数中要求给出变量地址，如给出变量名则会出错。如 scanf("%d",i)；是非法的，应改为 scanf("%d",&i)；才是合法的。

③ 在输入字符数据时，若格式控制串中无非格式字符，则认为所有输入的字符均为有效字符。例如：

```
scanf("%c%c ",&m, &n);
```

输入为：

```
d e
```

则把'd'赋予 m，' '赋予 n。只有当输入为：

```
de
```

时，才能把'd'赋予 m，'e'赋予 b。

如果在格式控制中加入空格作为间隔，如 scanf ("%c %c",&m,&n)；则输入时各数据之间可加空格。

例 3.6 scanf()函数使用示例。

```
/* 文件路径名:e3_6\main.c */
#include <stdio.h>                    /* 标准输入输出头文件 */
#include <stdlib.h>                   /* 包含库函数 system()所需要的信息 */

int main(void)                        /* 主函数 main() */
{
    char a, b;                        /* 定义变量 */
    printf("输入字符 a,b:");          /* 输入提示 */
    scanf("%c%c",&a, &b);             /* 输入 a, b */
    printf("%c%c\n",a, b);            /* 输出 a, b */

    system("PAUSE");                  /* 调用库函数 system(),输出系统提示信息 */
    return 0;                         /* 返回值 0, 返回操作系统 */
}
```

本例中 scanf()函数格式字符串"%c%c"中没有空格，输入 M N 时，a 的值为'M'，b 的