第4章　主菜单模块的开发

本章概述：

本章讲授了字符型数据和多分支 switch 语句的使用方法，学习本章后，读者应能够熟练运用多分支 switch 语句来解决问题。

教学重点：

- 字符型数据
- 多分支 switch 语句

教学难点：

- 多分支 switch 语句

解决方案：

通过分析小例程帮助读者理解 switch 语句的执行流程，通过精讲示例帮助读者掌握如何运用 switch 语句来解决实际问题。

在前面的章节中，一共开发了加法、减法、乘法、除法和求余共5个计算模块。本章将为计算器添加上主菜单的功能，使用户可以根据自己的需要选择相应的运算功能进行运算。

4.1 任务说明

任务描述：编写计算器中主函数(main)的菜单选择，输入不同的字母，进入不同的运算，如输入'A'，进入加法、输入'B'，进入减法等等。

任务要求：

(1)要求用户从键盘上输入一个大写字母。

(2)按照菜单的提示，输入相应的字母，进入对应的运算。

(3)输入的字母无效时，提示输入错误。

4.2 任务分析

☞ 开发思路

主菜单模块开发思路如下：

(1)用户根据菜单提示，输入一个字符选择相应的运算功能。

(2)程序根据用户选择的不同，调用与之相对应的运算模块。

☞ 新知识

1.字符类型

主菜单如下效果所示。

```
======= CACULATOR======
=                A 加法          =
=                B 减法          =
=                C 乘法          =
=                D 除法          =
=                E 求余          =
=                F 退出          =
=======================
```

在该菜单中，用户需要输入A、B等英文字母来完成功能的选择，在C语言中有一种专门用

来表示字符的数据类型——字符型，在定义该类型变量时用关键字 char 表示。定义一个字符型变量 choice 的语句为“char choice;”。

字符型变量在输入、输出时使用的格式说明符为%c，输入 choice 变量的语句为“scanf("%c",&choice);”。

由于字符型变量的特殊性，C 语言为字符型变量的输入提供了专门的函数，如 getchar 函数。所以我们也可以利用 getchar 函数来完成 choice 变量的输入。语句为“choice=getchar();”。

2.switch 多分支选择语句

由于本模块涉及到的分支较多，目前来看有 6 个，以后可能还要增加到 10 个以上，用 if-else 引导的分支语句会比较繁琐，所以这次使用 switch 引导的多分支选择语句。switch 语句与 if-else if 相比较，其特点是简单明了、结构清晰，适合条件简单、分支较多的分支语句。使用 switch 语句进行菜单选择的语句为：

```
switch(choice)
{
    case 'A' : add(); break;
    case 'B' : substract(); break;
    case 'C' : multiply(); break;
    case 'D' : divide(); break;
    case 'E' : complement(); break;
    case 'F' : exit(0); break;
    default : printf("输入错误! \n"); break;
}
```

4.3　任务实施

☞ 算法设计

根据上面的分析，给出该模块的 N-S 图如图 4-1 所示。

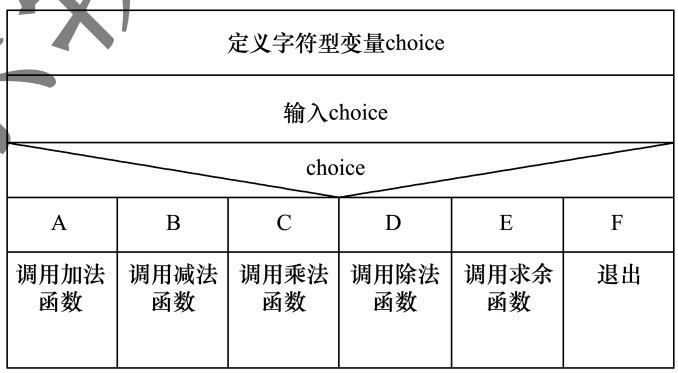


图 4-1　菜单选择的 N-S 图

☞ 算法实现

calculator.c 中的 main 函数

```
/*主菜单模块*/
#include<stdio.h>                          //包含所需要的预编译头文件 stdio.h
int main()                                 //主函数
{                                          /
    char choice;                           //定义选择变量 choice
    login();                               //调用口令验证函数 login()
    display_menu();                        //调用显示菜单函数 display_menu()
    printf("请输入选择:");
    choice=getchar();                      //输入选择
    switch(choice)                         //switch 引导的多分支选择语句,分支决策变
    {                                      //量为 choice
        case 'A' : add();break;            //choice==='A'时,调用加法函数
        case 'B' : substract();break;      //choice==='B'时,调用减法函数
        case 'C' : multiply();break;       //choice==='C'时,调用乘法函数
        case 'D' : divide();break;         //choice==='D'时,调用除法函数
        case 'E' : complement();break;     //choice==='E'时,调用求余函数
        case 'F' : exit(0);break;          //choice==='F'时,调用退出函数
        default : printf("输入错误! \n");break; //输入不正确的提示
    }
    return 0;
}
```

☞ 程序运行

```
请输入口令:123
密码正确,欢迎使用本系统!
=======CACULATOR======
=              A 加法          =
=              B 减法          =
=              C 乘法          =
=              D 除法          =
=              E 求余          =
=              F 退出          =
================== ==
请输入选择 A
请输入两个数 1 3
1+3=4
```

4.4　知识点详解

4.4.1　字符型数据

1.字符型常量

字符型常量是指仅含 ASCII 字符的常量,在内存中占一个字节,存放字符的 ASCII 码。字符常量的表示方法有两种:单引号表示法和转义字符表示法。

(1)单引号表示法。对于可显示的字符常量,可直接用单引号(特别注意:是半角的单引号)将该字符括起来,如'a','4','*','+'和'#'等。也可用字符的 ASCII 码值表示字符,如十进制数的 65 表示大写字母'A',八进制数的 0103 表示大写字母'C'。

(2)转义字符表示法。对于不能显示的字符(主要指控制字符,如回车符、换行符和制表符等)和一些在 C 语言中有特殊含义和用途的字符(如单引号、双引号、反斜杠线等),只能用转义字符表示。

转义字符是一种特殊的字符常量。转义字符由反斜线"\"开头,后面跟一个或几个字符。转义字符具有特定的含义,它不同于字符原有的意义,所以称"转义"字符。常用的转义字符及其含义如表 4-1 所示。

表 4-1　　常用的转义字符及其含义

转义字符	含义	ASCII 码
\n	换行	10
\r	回车	13
\f	换页	12
\t	水平制表(Tab)	9
\v	垂直制表	11
\b	退格符(backspace)	8
\\	反斜杠线"\"	92
\'	单引号符	39
\"	双引号符	34
\ddd	1～3 位八进制数所代表的字符	
\xhh	1～2 位十六进制数所代表的字符	
\a	报警响铃	7

C 语言字符集中的任何一个字符均可用转义字符来表示。表中的\ddd 和\xhh 正是为此而提出的。ddd 和 hh 分别为八进制和十六进制的数。如'\102'表示字母'B','\103'表示字母'C','\X0A'表示"换行"等。

转义字符主要用于控制数据的显示位置,如换行、换页、间隔长度等,巧妙地使用转义字符

可以使程序的输出更加整齐。

【例 4-1】 显示一个 3×3 矩阵。

程序清单 4-1 matrix.c

```
/*显示矩阵*/
#include<stdio.h>
int main()
{
    printf("\n1\t2\t3\n4\t5\t6\n7\t8\t9\n");
    return 0;
}
```

☞ **程序运行**

```
1 2 3
4 5 6
7 8 9
```

字符常量有如下的特点：

(1)字符常量只能用单引号括起来，不能用双引号或其他符号。

(2)字符常量只能是单个字符，不能是字符串。

(3)字符常量可以是字符集中的任意字符，但数字被定义为字符型之后就不能按原值参与数值运算。如'5'和 5 是不同的，'5'是字符常量，不能以数字 5 参与运算。

(4)C 语言中字符型和整数是不加区分的，字符型常量被视为占 1 个字节的整数，其值就是该字符的 ASCII 码，可以像整数一样参加数值运算。例如，'C'的 ASCII 码为 67，'C'－2 的值为 65，即字符'A'的 ASCII 码。

2.字符型变量

字符型变量用来存放字符常量，一个字符型变量只能存放一个字符，不要存放一个字符串。字符变量的类型说明符是 char。下面是几个字符型变量的定义：

```
char ch1,ch2,ch3;
```

上述语句将 ch1，ch2 和 ch3 定义为字符型变量，其内存可以各放一个字符，下面给这三个字符变量分别赋值'a'，'b'和'c'。

```
ch1='a';
ch2='b';
ch3='c';
```

【例 4-2】 字符型变量的定义与使用举例。

程序清单 4-2 char.c

```
/*字符型变量的定义与使用举例*/
#include<stdio.h>                    //包含所需要的预编译头文件 stdio.h
int main()                           //主函数
{
```

```
    char ch1,ch2 ;                          //定义两个字符型变量 ch1 和 ch2
    ch1=97 ; ch2=98 ;                       //分别为两个变量用整数赋值
    printf("%c, %c, ",ch1,ch2);             //用字符形式输出两个变量
    printf("%d, %d\n",ch1,ch2);             //用数值形式输出两个变量
    ch1=ch1-32;                             //ch1 自动转换成 ASCII 码值参与算术运算
    ch2=ch2-('a'-'A');                      //ch2 自动转换成 ASCII 码值参与算术运算
    printf("%c, %c\n",ch1,ch2);             //输出运算后的两个变量的字符形式
    return 0;
}
```

☞ 程序运行

```
a, b, 97, 98
A,B
```

☞ 编程点拨

在程序的开始处定义了两个字符型变量,然后分别将两个变量赋值为 97 和 98。

这里的 97 和 98 是字符'A'和'B'的 ASCII 码值,程序执行时 C 语言把 97 和 98 按照输出语句中格式控制符的规定转化成了其对应的字符。

接下来的两条输出语句以不同的格式输出 ch1 和 ch2,其中%c 表示以字符型格式输出 ch1 和 ch2,%d 表示以整型格式输出 ch1 和 ch2。

字符型数据和整型数据是通用的。它们既可以用字符形式输出,也可以用整数形式输出。但是应注意字符数据只占一个字节,只能存放 0～255 范围内的整数。

3.字符型数据的输入

字符型变量可以使用普通的输出函数 scanf 函数和专门用于字符输入的函数两种方法进行输入。

(1)使用 scanf 函数。使用 scanf 函数输出字符型时,使用“%c”格式说明符。如:“scanf("%c",&ch);”程序运行结果为:“A”,因为字符与 0～255 之间的整数有对应的关系,所以在此范围的整数可以以字符形式输出,同样字符也可以用整数的形式输出(输出的范围是 0～255)。

(2)使用专用于字符输入的函数。C 语言中的单个字符输入函数有三个,分别是:getchar、getche 和 getch。下面分别介绍。

①getchar 函数。getchar 函数是一个不带参数的字符输入函数。其功能是从标准输入设备输入一个字符,调用格式如下:

```
getchar();
```

程序中用到 getchar 函数时要在程序的开头部分用以下的命令:

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

【例 4-3】 getchar 函数应用举例。

程序清单 4-3 exp_getchar.c

```
/*格式控制符%c应用举例*/
#include<stdio.h>
int main()
{
    int n;
    printf("Please input a character:\n");
    n=getchar();
    printf("n=%d\n",n);
    return 0;
}
```

☞ 程序运行

```
Please input a character:
a
n=65
```

☞ 编程点拨

用 getchar 函数从键盘上接收的字符既可以是打印的字符，也可以是非打印的字符，但从键盘敲不出来的字符除外。由于系统存在“仿效返回”，当用户在输入数据时，系统会马上显示出相应的字符，这个字符不是程序的输出，而是系统的仿效返回。一般要在键入一个回车键之后，再次显示的字符才是程序执行的。尽管 getchar 函数只接收一个字符，但实际上，用户键入回车键以后，系统才开始接收字符。因此运行上述程序时，输入'A'和"ABC"，输出结果是相同的。

②getche 函数和 getch 函数。getche 函数和 getch 函数的功能是从标准输入设备输入一个字符，与 getchar 不同的是：getchar 输入一个字符后必须按回车键才能被接收，而 getche 和 getch 输入字符后不必按回车键，其中 getch 不回显输入字符。

getche 和 getch 在头文件“conio.h”文件中定义。

4.字符型数据的输出

除了第 1 章讲过的 printf 函数以外，C 语言还提供 putchar 函数用于单个字符输出。两者在使用时格式和功能上都有很大的区别。

(1)使用 printf 函数输出字符，其格式说明符使用%c，如：“printf("%c",ch);”。

因为字符与 0～255 之间的整数有对应的关系，所以在此范围的整数可以以字符形式输出，同样，字符也可以用整数的形式输出(输出的范围是 0～255)。

【例 4-4】 格式控制符“%c”应用举例。

程序清单 4-4 printf_c.c

```
/*格式控制符%c应用举例*/
#include<stdio.h>
```

```
int main()
{
    int n=65;
    char ch='A';
    printf("\n n= %d, %c",n,n);
    printf("\n ch= %d, %c",ch,ch);
    return 0;
}
```

☞ **程序运行**

```
n=65,A
ch=65,A
```

(2)使用 putchar 函数输出字符。

putchar 函数调用的一般格式如下：

```
putchar(ch);
```

其中 ch 是一个字符型常量或变量，也可以是一个不大于 255 的整型常量或变量。该函数的功能是向标准输出设备(一般指显示器终端)输出一个字符。

程序中用到 putchar 函数时要在程序的开头部分使用下列包含命令：

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

用 putchar 函数输出字符时，只需把待输出的字符作为其参数即可。用 putchar 输出时有以下几种方式：

①输出字符变量。

```
char ch ='A';
putchar(ch);
```

②输出字符常量。

```
putchar('A');
putchar(65);
```

③输出不可见字符。

```
putchar('007');          /*输出响铃*/
putchar(007);            /*输出响铃*/
putchar('\n');           /*输出换行*/
```

4.4.2　多分支 switch 语句

1.语句格式

switch 语句是多分支选择语句。if 语句一般适用于两个分支供选择的情况，即在两个分支中选择其中一个执行。尽管可以通过 if 语句的嵌套形式实现多路选择的目的，但这样做的结果使得 if 语句的嵌套层次太多，降低了程序的可读性。C 语言中的 switch 语句，提供了更加清晰、方便地进行多分支选择的功能。

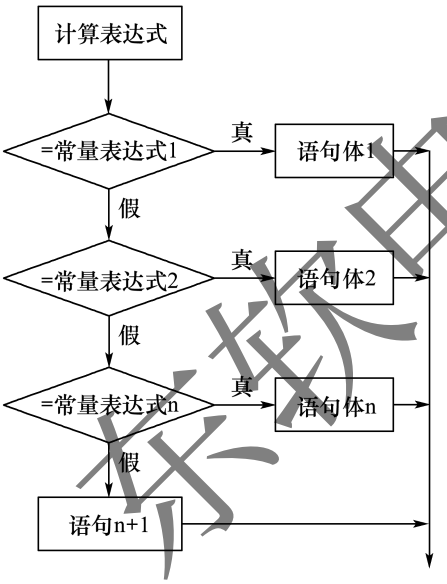
switch 语句的一般形式如下所示：

```
switch(表达式)
```

```
{
    case  常量表达式 1:
          语句体 1;
          [break];
    case  常量表达式 2:
          语句体 2;
          [break];
    ...
    case  常量表达式 n:
          语句体 n;
          [break];
    default: 语句 n+1;
}
```

其语义为:计算表达式的值,然后逐个和 case 后面的常量表达式值相比较,当表达式的值与某个 case 后面的常量表达式值相等时,则执行其后的语句体;如果都不相等,则执行 default 后面的语句。如果没有 default 部分,则不执行 switch 语句中的任何语句,而直接转到 switch 语句后面的语句去执行。

switch 语句的流程图和 N-S 图如图 4-2 所示。



(a)switch语句的流程图

计算表达式				
常量表达式1	常量表达式2	……	常量表达式n	default
语句体1	语句体2	……	语句体n	语句体n+1

(b)switch语句的N-S图

图 4-2

2.格式说明

(1)switch 语句后面圆括号内表达式的值和 case 后面的常量表达式的值,都必须是整型或字符型的,不允许是其他类型。

(2)同一个 switch 语句中的所有 case 后面的常量表达式的值都必须互不相同。

(3)switch 语句中的 case 和 default 的出现次序是任意的,也就是说 default 也可以位于

case 后面，且 case 的次序也不要求按常量表达式的大小顺序排列。

(4)default 和“语句 n+1”可以同时省略。

(5)由于 switch 语句中的“case 常量表达式”部分只起语句标号的作用，而不进行条件判断，所以，在执行某个 case 后面的语句后，将自动转到该语句后面的语句去执行，直到遇到 switch 语句的右大括号或“break”语句为止，而不再进行条件判断。例如：

```
switch(n)
{
    case 1 : x=1;
    case 2 : x=2;
}
```

当 n=1 时，将连续执行下面两个语句：

```
x=1;
x=2;
```

所以在执行完一个 case 分支后，一般应跳出 switch 语句，转到下一条语句执行，这样可在一个 case 结束后，下一个 case 开始前，插入一个 break 语句。一旦执行到 break 语句，将立即跳出 switch 语句，例如：

```
switch(n)
{
    case 1 : x=1;
        break;
    case 2 : x=2;
        break;
}
```

(6)每个 case 的后面既可以是一条语句，也可以是多条语句，当是多条语句的时候，也不需要用大括号括起来。

(7)多个 case 的后面可以共用一组执行语句。例如：

```
switch(n)
{
    case 1 :
    case 2 : x=2;
        break;
    ...
}
```

它表示当 n=1 或 n=2 时，都执行下面两个语句：

```
x=2;
break;
```

【例 4-5】 从键盘输入一个星期数(0～6)，显示该星期的英文名称。

程序清单 4-5 week.c

```
/*switch语句举例*/
#include<stdio.h>                    //包含所需要的预编译头文件 stdio.h
```

```
int main()                                    //主函数
{
    int WeekDay;
    printf("\n Please input the week day: ");  //提示信息
    scanf("%d",&WeekDay);                      //从键盘输入一个星期数
    switch(WeekDay)
    {
        case 0 : printf("\n Sunday\n"); break;
        case 1 : printf("\n Monday\n"); break;
        case 2 : printf("\n Tuesday\n"); break;
        case 3 : printf("\n Wednesday\n"); break;
        case 4 : printf("\n Thursday\n"); break;
        case 5 : printf("\n Friday\n"); break;
        case 6 : printf("\n Saturday\n"); break;
        default : printf("\n Input Error! \n"); break;
    }
    return 0;
}
```

☞ **程序运行**

```
Please input the week day:6
Saturday
```

自测题

1. 已知字符'A'的ASCII码为65,则下列程序的输出结果是_____。

```
#include<stdio.h>
int main()
{
    char c1=65,c2=66;
    printf("%d %c",c1,c2);
    return 0;
}
```

2. 以下程序的执行结果为_____。

```
#include<stdio.h>
int main()
{
    char c='A'+10;
    printf("c= %c\n",c);
    return 0;
}
```

3. 编写程序,从键盘输入两个数字字符并分别存放在字符型变量x和y中,要求通过程序

将这两个字符对应的数字相加后输出。

4. 编写程序，从键盘上输入一个小写字母，将其转化为大写字母。

5. The following program takes user input a student's score and display a level in the next line on the screen, but it always displays the wrong result. Determine where the errors are and why the program can never give the correct result.

```
int mian()
{
    int score;
    char grade;
    printf("input a score(0～100)\n");
    scanf("%d", &score);
    switch (score)
    {
        case 0:case 1:case 2:case 3:case 4: case 5:
            printf("grade=E\n");
        case 6:
            printf("grade=D\n");
        case 7:
            printf("grade=C\n");
        case 8:
            printf("grade=B\n");
        case 9:
            printf("grade=A\n");
        default:
            printf("The score is out of range! \n");
    }
    return  0;
}
```