

第3章 选择结构程序设计

选择结构要根据某个条件的满足与否,决定执行不同的操作。实现选择结构有 if 和 switch 两种语句。

3.1 if 语句

if 语句有 3 种形式,即单分支 if 语句、双分支 if 语句和多分支 if 语句。

3.1.1 关系运算与单分支 if 语句

单分支 if 语句的一般形式是:

if(表达式) 内嵌语句

其中,表达式可以是值为数值型的任何表达式,当然也可以是表达式的特殊形式——常量、变量或函数。表达式值非 0 即为真,值为 0 就是假。括号后的语句是 if 语句的内嵌语句,注意只能内嵌一个语句。其执行过程见图 3.1。

例如:

```
if(x>y) max=x;
```

执行时先判断 x 是否大于 y。若 $x>y$,则将 x 的值赋给 max,然后执行下一条语句;否则,直接执行下一条语句。

语句中的 $x>y$ 是一个**关系表达式**,即用关系运算符将两个运算对象连接起来的合法的式子。**关系运算符**有 6 个:①<(小于);②<=(小于或等于);③>(大于);④>=(大于或等于);⑤==(等于);⑥!=(不等于)。

关系运算符都是双目运算符,左结合,其优先级高于赋值运算符,低于算术运算符。前 4 种关系运算符<、<=、>和>=的优先级相同,后两种==和!=的优先级相同,而前 4 种的优先级高于后两种。

关系表达式的值只有两个。关系表达式成立时,表示真,其值为 1;关系表达式不成立时,表示假,其值为 0。例如,若 $a=3, b=2, c=1$,则 $a>b$ 的值为 1, $a+b==c$ 的值为 0, $b<1$ 的值为 0。

特别地, $a>b>c$ 的值是 0,这是由于两个>优先级相同,结合方向为自左至右,故先做 $a>b$,其值为 1,然后做 $1>c$,值为 0。所以,要表达 a、b、c 满足严格递降关系,即数学意义上的 $a>b>c$ 时,在 C 程序中直接写成 $a>b>c$ 是错误的,应该使用 3.1.3 节介绍的逻辑表达式。

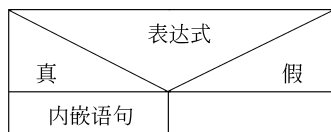


图 3.1 单分支 if 语句执行过程



关系运算



单分支 if 语句

【例 3-1】 输入一个数,若为正数则输出该数。

```
#include "stdio.h"
main()
{   int a;
    scanf("%d", &a);
    if(a>0) printf("%d\n", a);           /* 如果 a 是正数就输出 a 的值 */
}
```

运行结果:

6 ✓
6

再运行一次,其结果:

-3 ✓

【例 3-2】 对输入的两个实数按由小到大的顺序输出。

```
#include "stdio.h"
main()
{   float a,b;
    scanf("%f%f", &a, &b);
    if(a<=b) printf("%f, %f\n", a, b);    /* 如果 a 比较小,就先输出 a,后输出 b */
    if(a>b) printf("%f, %f\n", b, a);    /* 如果 b 比较小,就先输出 b,后输出 a */
}
```

运行两次的结果:

1 2 ✓
1.000000, 2.000000
2 1 ✓
1.000000, 2.000000

若希望输出时一定按 a、b 的顺序,则在输出之前必须保证 a 中存放两者中较小的数, b 中存放较大的数。如果输入的两数中 a 较大,则必须交换 a、b 的值。

如何实现呢? 可以这样想: a、b 是两盒磁带, a 中是歌曲, b 中是英语, 若希望交换这两盒磁带的内容, 必须再有一盒空白带 t。可以先将磁带 a 中的歌曲转录到空白带 t 中 ($t=a$), 然后将磁带 b 中的英语录到磁带 a 中 ($a=b$), 最后将磁带 t 中的歌曲录到磁带 b 中 ($b=t$), a、b 的内容就互换了。也就是说, 如果 $a>b$, 就执行 3 个语句 $t=a; a=b; b=t;$ 。但是, if 语句只能内嵌一个语句, 所以必须用 {} 把这 3 个语句括起来构成一个 **复合语句**。该方法程序如下。

```
#include "stdio.h"
main()
{   float a,b,t;
    scanf("%f%f", &a, &b);
```

```

if(a>b)          /* 如果 a>b,就交换 a 和 b 的值 */
{
    t=a;
    a=b;
    b=t;
}
printf("%f,%f\n",a,b);
}

```

可以看到,内嵌语句从 if(表达式)的下一行开始书写时,内嵌语句的各行都整齐地向后缩进了若干列。建议采用此种缩格方式书写程序,既便于读懂程序,也便于查找错误。

程序中的 if 语句也可以改为

```
if(a>b) t=a,a=b,b=t;
```

其中的逗号不是分隔符,而是**逗号运算符**,用来把两个或多个表达式连接起来,组成一个逗号表达式。

逗号表达式的一般形式是:

表达式 1,表达式 2,...,表达式 n

逗号运算符也称**顺序求值运算符**,是双目运算符,优先级最低,结合方向是自左至右。逗号表达式的运算过程是按从左到右的顺序逐个计算每一个表达式,即先计算表达式 1,再计算表达式 2,……,最后计算表达式 n。逗号表达式的值和类型是最后一个表达式的值和类型。

在修改后的 if 语句中,由于使用了逗号运算符,括号后是一条语句,就不需要用{}把它们括起来了。逗号运算符常用于此种情况,往往并不关心逗号表达式的值。

3.1.2 求余运算与双分支 if 语句

双分支 if 语句的一般形式是:

```

if(表达式) 语句 1
else 语句 2

```

双分支 if 语句的执行过程见图 3.2。双分支 if 语句是一条语句,它先计算表达式的值,若表达式值为非 0(即为真),则执行语句 1;若表达式值为 0(即为假),则执行语句 2。执行语句 1 或语句 2 后,都直接执行 if 语句的下一条语句。

【例 3-3】 输入一个整数,若为偶数则输出 Yes,若为奇数则输出 No。

【分析】 判断一个整数的奇偶性,需使用求余运算符%。 $a\%b$ 的结果是 a 除以 b 的余数。例如, $5\%2$ 的值是 1, $8\%2$ 的值是 0, $2\%5$ 的值是 2。**求余运算符**也是基本算术运算符,优先级与乘法、除法运算符相同,结合方向为自左至右。参与求余运算的两个量必须都是整型的。

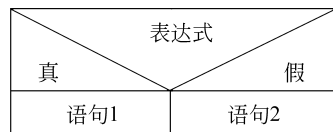


图 3.2 双分支 if 语句执行过程



双分支
if 语句

对于一个整数 a , 可以根据 $a\%2$ 的值是否为 0 来判断其奇偶性。

```
#include "stdio.h"
main()
{   int a;
    scanf("%d", &a);
    if(a%2==0) printf("Yes\n");      /* 如果 a 是偶数就输出 Yes */
    else      printf("No\n");       /* 否则(a 是奇数)输出 No */
}
```

运行两次的结果:

```
5 ✓
No
12 ✓
Yes
```

【说明】 程序中的 if 语句可以改为

```
if(a%2) printf("No\n");      /* 如果 a 是奇数就输出 No */
else    printf("Yes\n");     /* 否则(a 是偶数)输出 Yes */
```

如果 a 是奇数, 则 $a\%2$ 的值为 1, 非 0, 将输出 No; 如果 a 是偶数, 则 $a\%2$ 的值为 0, 将输出 Yes。

【注意】 求余运算的两个操作数必须都是整型的。

例 3-2 也可以用双分支 if 语句实现。

```
#include "stdio.h"
main()
{   float a,b;
    scanf("%f%f", &a, &b);
    if(a<=b) printf("%f, %f\n", a, b);    /* 如果 a 比较小, 就先输出 a, 后输出 b */
    else     printf("%f, %f\n", b, a);    /* 否则(b 比较小)先输出 b, 后输出 a */
}
```



3.1.3 逻辑运算与多分支 if 语句

多分支
if 语句

多分支 if 语句的一般形式是:

```
if(表达式 1) 语句 1
else if(表达式 2) 语句 2
...
else if(表达式 m) 语句 m
[else 语句 m+1]
```

其中的方括号表示其中内容可以缺省。

多分支 if 语句执行过程如图 3.3 所示。

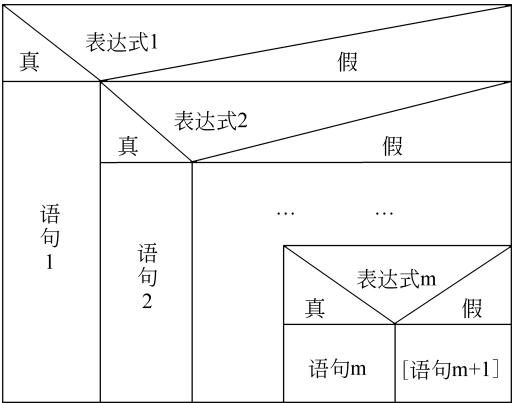


图 3.3 多分支 if 语句执行过程

多分支 if 语句是一条语句，它首先计算表达式 1 的值是否非 0，非 0 即为真，就执行语句 1，整个 if 语句结束；否则计算表达式 2 的值是否非 0，非 0 即为真，就执行语句 2，整个 if 语句结束；否则计算表达式 3 的值是否非 0，……，如果所有 m 个表达式的值都为 0，即都为假，有 else 就执行 else 后面的语句 m+1，整个 if 语句结束；没有 else 则整个 if 语句直接结束。

在有 else 的多分支 if 语句中，只有一个内嵌的语句会被执行到，即多个分支中只能执行一个。不管执行了哪一个分支后都会直接执行多分支 if 语句的下一条语句。在无 else 的多分支 if 语句中，多个分支可能执行一个，然后执行多分支 if 语句的下一条语句；也可能一个也不执行，直接执行多分支 if 语句的下一条语句。

【注意】 else 总是与它上面离它最近的尚未有 else 与之对应的 if 对应。else 不能单独使用，它只能出现在双分支或多分支 if 语句中，且必须有与之对应的 if。

【例 3-4】 函数 $y = \begin{cases} -1, & x < 0 \\ 0, & x = 0, \text{输入 } x \text{ 的值后, 请输出相应的 } y \text{ 值。} \\ 1, & x > 0 \end{cases}$

【分析】 该分段函数的定义域覆盖了整个实数轴，故可用带有 else 的三分支 if 语句来实现。

```
#include "stdio.h"
main()
{ int x,y;
  scanf("%d",&x);
  if (x<0) y=-1;          /* 如果 x<0, 则 y 赋值为 -1 */
  else if (x==0) y=0;      /* 否则如果 x=0, 则 y 赋值为 0 */
  else y=1;               /* 否则 (x>0) y 赋值为 1 */
  printf("%d\n",y);
}
```

运行两次的结果:

```
3 ✓
1
0 ✓
0
```

【说明】 从分段函数的定义来看, x 用浮点型比较合适, 但 y 还应该是整型。读者可以修改程序, 把 x 用浮点型表示。

【注意】 判断两个数是否相等, 必须使用关系运算符 $==$ 。

【例 3-5】 函数 $z = \begin{cases} -1, & x < 0 \text{ 且 } y < 0 \\ 0, & x = 0 \text{ 或 } y = 0, \text{ 输入 } x, y, \text{ 输出相应的 } z \text{ 值。} \\ 1, & x > 0 \text{ 且 } y > 0 \end{cases}$

如何表达两者之间的“并且”“或者”关系呢? 这就要用到逻辑运算符。逻辑运算符有 3 个: $\&\&$ (逻辑与)、 $\|\|$ (逻辑或) 和 $!$ (逻辑非)。

逻辑与($\&\&$)表示“并且”, 当 a 和 b 都非 0(为真)时, $a\&\&b$ 的值为 1, 否则值为 0。逻辑或($\|\|$)表示“或者”, a 和 b 中只要有一个非 0, $a\|\|b$ 的值就是 1; 只有二者都是 0 时, $a\|\|b$ 的值才是 0。逻辑非(!)表示“非”, a 非 0 时, $!a$ 的值是 0; a 为 0 时, $!a$ 的值是 1。

$!$ 是单目运算符。单目运算符的优先级都相同, 结合方向都是自右至左。 $!$ 的优先级与自增、自减运算符相同, 高于基本算术运算符。 $\&\&$ 和 $\|\|$ 都是双目运算符, 左结合, 优先级低于关系运算符, 高于赋值运算符。 $\&\&$ 的优先级高于 $\|\|$ 。

用逻辑运算符连接的合法的式子称为逻辑表达式。逻辑表达式的值应该是一个逻辑量真或假。

【注意】 C 语言给出逻辑运算结果时, 以数值 1 表示真, 以 0 表示假; 但判断一个量是否为真时, 以 0 表示假, 以非 0 表示真。

若 $a=3, b=5$, 则 $!a$ 值为 0, $a\&\&b$ 值为 1, $a\|\|b$ 值为 1, $!a\&\&b$ 值为 0, $a-b\&\&0\|\|0$ 值为 0。

学习了逻辑运算符和逻辑表达式后, 就能够表达 a, b, c 满足严格递降关系, 即数学意义上的 $a > b > c$ 了, 例如, 可以写作 $a > b \&\& b > c$ 。也能编写出例 3-5 的程序了。

```
#include "stdio.h"
main()
{ float x,y;int z;
  scanf("%f%f",&x,&y);
  if (x<0&&y<0) z=-1; /* 如果 x<0 且 y<0, z 赋值为 -1 */
  else if (x==0||y==0) z=0; /* 否则如果 x=0 或 y=0, z 赋值为 0 */
  else if (x>0&&y>0) z=1; /* 否则如果 x>0 且 y>0, z 赋值为 1 */
  printf("%d\n",z); /* 第 8 行 */
}
```

运行 3 次的结果:

```
-1 -5 ✓
```



逻辑运算

```

-1
0 0 ✓
0
1 -6 ✓
-858993460

```

【说明】 为什么最后一次的运行结果会是这样呢？这是由于 x 的值是 1、 y 的值是 -6，不在该分段函数的定义域内。执行 `if` 语句时，3 个表达式的值都是 0，于是没有执行任何一个分支，就直接执行输出 z 的值了。由于在程序的执行过程中 z 没有被赋值，于是就输出了一个不确定的值。

为了避免产生这种错误，可以只对定义域内的 x 、 y 输出 z 值，而对定义域外的 x 、 y 输出数据错误的信息。例如，第 8 行语句可以改为

```

if (x * y >= 0) printf("%d\n", z); /* 如果 x、y 不异号 (在定义域内)，就输出 z 的值 */
else printf("数据错误\n");      /* 否则输出“数据错误” */

```

【注意】 在一个变量没有得到确定的值之前，是不可以使用它的值的。

3.1.4 if 语句的嵌套

if 语句中内嵌了一个或多个 if 语句称为 **if 语句的嵌套**。例如：

```

if()
    if() 语句 1
    else 语句 2
else
    if() 语句 3      内嵌 if 语句

```

} 内嵌 if 语句

也可以用 if 语句的嵌套编写出例 3-4 程序如下。

```

#include "stdio.h"
main()
{ int x, y;
  scanf("%d", &x);
  if (x <= 0)                /* 如果 x ≤ 0 */
      if (x < 0) y = -1;     /* 如果 x < 0, y 赋值为 -1 */
      else y = 0;           /* 否则 (x = 0), y 赋值为 0 */
  else y = 1;                /* 否则 (x > 0) y 赋值为 1 */
  printf("%d\n", y);
}

```



if 语句
的嵌套

3.1.5 条件运算符与条件表达式

条件运算符由?和:两个符号组成，是 C 语言中唯一的一个三目运算符，右结合，优先



条件运算

级低于逻辑运算符,高于赋值运算符。

条件表达式的一般形式是:

表达式 1 ? 表达式 2 : 表达式 3

如果表达式 1 的值非 0,则条件表达式的值是表达式 2 的值,否则条件表达式的值是表达式 3 的值。

例如, $\text{max}=(a>b)?a:b$; 相当于:

```
if (a>b) max=a;
else    max=b;
```

3.1.6 程序举例



程序举例

【例 3-6】 有如下的分段函数,输入 x 的值后,请输出相应的 y 值。

$$y = \begin{cases} \frac{5}{29}|x-7|, & x < -10 \\ \log_3 16 + \cos 32^\circ, & -10 \leq x < 12.6 \\ \frac{\sqrt{2x} - \pi \sin x}{e^{x^2}}, & x \geq 12.6 \end{cases}$$

【分析】 该分段函数的定义域覆盖了整个实数轴,故可用带有 else 的三分支 if 语句来实现。由分段函数的定义可知, x 、 y 用浮点型比较恰当。由于程序中用到了数学函数,故需要使用文件包含命令 `#include "math.h"`。

```
#include "stdio.h"
#include "math.h"
main()
{ float x,y;
  scanf("%f",&x);
  if(x<-10) y=5.0/29*fabs(x-7);           /* 第 6 行 */
  else if(x<12.6) y=log(16)/log(3)+cos(32*3.14159/180); /* 第 7 行 */
  else y=(sqrt(2*x)-3.14159*sin(x))/(exp(1)*x*x); /* 第 8 行 */
  printf("%f\n",y);
}
```

运行 3 次的结果:

```
-15.6 ✓
3.896552
9 ✓
3.371767
68.7 ✓
0.001012
```

【说明】 该程序中函数的定义域是 $(-\infty, +\infty)$,而且各分支是按照 x 从小到大的顺序,所以程序若能执行到计算第二个表达式 $x < 12.6$ 的值,一定是第一个表达式 $x < -10$ 的值为假,即 $x \geq -10$,所以在第二个表达式中没有判断 x 是否大于或等于 -10 ;同理,若能执行到 `else` 子句,一定是前两个表达式的值均为假,即 $x \geq 12.6$,所以就不用再进行判断了。当然,在每个分支都进行所有判断也是可以的,即程序中的 `if` 语句(第 6~8 行)也可以改为

```
if(x < -10) y = 5.0/29 * fabs(x - 7);
else if(x >= -10 && x < 12.6) y = log(16)/log(3) + cos(32 * 3.14159/180);
else if(x >= 12.6) y = (sqrt(2 * x) - 3.14159 * sin(x)) / (exp(1) * x * x);
```

如果函数的定义域不是 $(-\infty, +\infty)$,或者各分支没有按照自变量从小到大或者从大到小的顺序排,则必须在每个分支都进行所有判断。

该程序中涉及多个算术表达式。

【注意】 在书写算术表达式时,

- (1) 乘号不能省。例如, $2x$ 必须写成 $2 * x$ 。
- (2) 分子、分母若是表达式,一般应将其用括号括起来,避免出错。例如, $\frac{x+y}{2a}$ 应写作 $(x+y)/(2*a)$ 。
- (3) 两个整型量相除结果仍为整型,所以至少要将其中的一个转换为浮点型量。例如, $\frac{5}{29}$ 可以写作 $5.0/29$,系统会自动将 29 转换成浮点型 29.0,然后再进行运算。
- (4) 函数的自变量必须用括号括起来。例如, $\sin x$ 必须写作 $\sin(x)$ 。
- (5) 表达式中只可使用圆括号。可以多层嵌套使用,但左、右括号必须匹配。
- (6) 三角函数自变量的单位是弧度,若为角度必须转换为弧度。例如, $\cos 32^\circ$ 可以写作 $\cos(32 * 3.14159/180)$ 。
- (7) 若所用对数函数既不是自然对数也不是常用对数,则利用换底公式将其用自然对数或常用对数来表示。例如, $\log_3 16$ 可以写作 $\log(16)/\log(3)$ 或 $\log_{10}(16)/\log_{10}(3)$ 。
- (8) 使用自然对数的底数 e 时必须写作 $\exp(1)$,因为直接写作 e 系统会认为是一个变量; π 只能用它的某个近似值来表示,例如可以写作 3.14159。

【例 3-7】 有如下的分段函数,输入 x 的值后,请输出相应的 y 值。

$$y = \begin{cases} x, & 15 \leq x < 30 \\ 50, & 30 \leq x < 100 \\ 2x - 3, & 100 \leq x < 200 \\ \text{无意义}, & \text{其他} \end{cases}$$

【分析】 该函数在定义域 $[15, 200)$ 内可以根据其所在位置得到相应的 y 值,但在定义域外只能输出无意义。因此,应首先使用一个双分支 `if` 语句,区分开 x 是在定义域内还是在定义域外。若 x 在定义域外,则直接输出“无意义”;若 x 在定义域内,则再使用一个多分支 `if` 语句,根据 x 所在位置得到相应的 y 值,然后再输出该值。

```

#include "stdio.h"
main()
{ float x,y;
  scanf("%f",&x);
  if(x>=15&&x<200)          /* 如果 x 在定义域内 */
  { if(x<30) y=x;           /* 计算 y 的值 */
    else if(x<100) y=50;
    else y=2*x-3;
    printf("y=%.2f\n",y);    /* 输出 y 的值 */
  }
  else                       /* 否则 (x 不在定义域内) */
    printf("无意义\n");      /* 输出 "无意义" */
}

```

运行两次的结果:

```

62.5 ✓
y=50.00
3 ✓
无意义

```

【说明】 当 x 在定义域内时,要执行两个语句,所以必须使用复合语句,即用 $\{ \}$ 将这两个语句括起来。

3.2 switch 语句



switch 语句

实现多分支也可以使用 switch 语句。

switch 语句的一般形式是:

```

switch(表达式)
{
    case 常量 1: 若干语句
    case 常量 2: 若干语句
    ...
    case 常量 m: 若干语句
    default: 若干语句
}

```

其中,表达式的值必须是整型或字符型。执行 switch 语句时首先计算 switch 后面括号中表达式的值。若表达式的值与某个 case 后面常量的值相等,就从该 case 后面的语句开始执行,直到 switch 语句的结束;如果表达式的值与所有 case 后面常量的值都不相等,就执行 default 后面的语句。

default 子句可以缺省。若表达式的值与所有 case 后面常量的值都不相等,而且没有 default 子句,则 switch 语句就直接结束。

【例 3-8】 输入学生的成绩(分数),输出其对应的等级。优(90~100)、良(80~89)、中(70~79)、及格(60~69)和不及格(0~59)分别用 A、B、C、D、E 表示。

【分析】 考虑从成绩的十位数容易得出其对应的等级,于是可以先做 $x/10$,然后取它的整数部分,9、10 为优,8 为良,7 为中,6 为及格,其余为不及格。

取一个实数的整数部分可以使用强制类型转换运算符实现。

强制类型转换的一般形式是:

(类型标识符)(表达式)

其中(类型标识符)是强制类型转换运算符,作用是将其后面表达式的值转换为括号中指定的类型。例如,浮点型变量 a 的值是 3.6,则 $(\text{int})a$ 的值是 3, $(\text{int})(a-1)$ 的值是 2, $(\text{float})(2*3)$ 的值是 6.0。强制类型转换运算符也是算术运算符,它是单目运算符,所以优先级高于基本算术运算符,结合方向是右结合。

【注意】 强制类型转换只是转换表达式值的类型,对所涉及变量的类型和值没有影响。

例如,浮点型变量 a 的值是 3.6,则表达式 $(\text{int})a$ 的值是 3,但运算后 a 的类型仍是浮点型,值仍为 3.6,就像做运算 $2*a$ 后 a 的类型和值都不变一样。

利用强制类型转换运算符编写例 3-8 程序如下。

```
#include "stdio.h"
main()
{ float x;
  int y;
  scanf("%f",&x);
  y=(int)(x/10);
  switch(y)
  { case 10:
    case 9: printf("A\n");
    case 8: printf("B\n");
    case 7: printf("C\n");
    case 6: printf("D\n");
    default: printf("E\n");
  }
}
```

运行结果:

```
86 ✓
B
C
D
E
```

【说明】 为什么没有得到正确的结果呢?这是由于 switch 后面括号中表达式 y 的值

是 8,于是从 case 8 后面的语句开始执行,直到 switch 语句结束,即从语句 printf("B\n"); 开始执行,直到 printf("E\n");结束。那么,如何在执行一个 case 分支后,终止 switch 语句的执行呢? 可以使用 break 语句来达到此目的。修改后的正确程序如下:

```
#include "stdio.h"
main()
{ float x;
  int y;
  scanf("%f",&x);
  y=(int)(x/10);
  switch(y)
  { case 10:
    case 9: printf("A\n"); break;
    case 8: printf("B\n"); break;
    case 7: printf("C\n"); break;
    case 6: printf("D\n"); break;
    default: printf("E\n");
  }
}
```

运行结果:

```
86 ✓
B
```

【注意】

- (1) case 后面冒号之前的部分必须是常量或常量表达式,不能出现变量或函数。
- (2) 各 case 后面常量的值必须互不相同。
- (3) 多个 case 可以共用一组语句。如例 3-8 中的

```
case 10:
case 9: printf("A\n");break;
```

本章小结

本章介绍了实现选择结构的 if 语句和 switch 语句,重点是 if 语句。if 语句有 3 种形式,分别是单分支 if 语句、双分支 if 语句和多分支 if 语句。

本章还介绍了多种运算符及其对应的表达式,并在 3.1.6 节中重点介绍了书写算术表达式时的注意事项。

目前所介绍的运算符中,按优先级由高到低分别为:

- (1) 括号()。
- (2) 单目运算符++、--、!、(类型标识符)。
- (3) 基本算术运算符*、/、%、+、- ,其中*、/和%的优先级高于+、-。

(4) 关系运算符 $>$ 、 $>=$ 、 $<$ 、 $<=$ 、 $=$ 、 $!=$ ，其中 $>$ 、 $>=$ 、 $<$ 和 $<=$ 的优先级高于 $=$ 、 $!=$ 。

(5) 逻辑与 $\&$ 。

(6) 逻辑或 $\|$ 。

(7) 条件运算符 $?:$ 。

(8) 赋值运算符 $=$ 。

(9) 逗号运算符。

单目运算符和唯一的三目运算符 $?:$ 的结合方向都是自右至左，双目运算符中只有赋值运算符的结合方向是自右至左，其余双目运算符的结合方向都是自左至右。

习题 3

3-1 输入一个学生的成绩 $score$ ，如果 $score \geq 60$ ，则输出 $pass$ ，否则输出 $fail$ 。

3-2 输入两个数，输出其中较大的数。

3-3 输入 3 个数，对其按照由小到大的顺序输出。

3-4 输入 3 条边长 a 、 b 、 c ，如果它们能构成一个三角形就计算该三角形的面积，否则输出不是三角形的信息。

3-5 有如下的分段函数，输入 x 的值后，请输出相应的 y 值。

$$y = \begin{cases} \log_2 3 + x \sin 66^\circ, & x < 3 \\ e, & x \geq 3 \end{cases}$$

3-6 有如下的分段函数，输入 x 的值后，请输出相应的 y 值。

$$y = \begin{cases} \frac{1}{6}e^x + \sin x, & x > 1 \\ \sqrt{2x+5}, & -1 < x \leq 1 \\ \frac{|x+3|}{x^3-7}, & x \leq -1 \end{cases}$$

3-7 有如下的分段函数，输入 x 的值后，请输出相应的 y 值。

$$y = \begin{cases} \cos x, & 1 < x \leq 6 \\ 3x, & 12 < x \leq 27 \\ \text{无意义}, & \text{其他} \end{cases}$$

3-8 输入 1~7 中的任意一个数字，输出对应的星期几的英文单词。