5　分支结构程序.....1
5.1　关系运算符和表达式.....1
5.1.1　关系运算符及其优先次序.....1
5.1.2　关系表达式.....1
5.2　逻辑运算符和表达式.....2
5.2.1　逻辑运算符极其优先次序.....2
5.2.2　逻辑运算的值.....3
5.2.3　逻辑表达式.....3
5.3　if 语句.....4
5.3.1　if 语句的三种形式.....4
5.3.2　if 语句的嵌套.....7
5.3.3　条件运算符和条件表达式.....9
5.4　switch 语句.....10
5.5　程序举例.....11

5 分支结构程序

1.1 关系运算符和表达式

在程序中经常需要比较两个量的大小关系，以决定程序下一步的工作。比较两个量的运算符称为关系运算符。

1.1.1 关系运算符及其优先次序

在C语言中有以下关系运算符：

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，其结合性均为左结合。关系运算符的优先级低于算术运算符，高于赋值运算符。 在六个关系运算符中，<, <=, >, >=的优先级相同，高于==和!=，==和!=的优先级相同。

1.1.1 关系表达式

关系表达式的一般形式为：

表达式 关系运算符 表达式

例如：

a+b>c-d

x>3/2

'a'+1<c

-i-5*j==k+1

都是合法的关系表达式。由于表达式也可以又是关系表达式。 因此也允许出现嵌套的情况。例如：

a>(b>c)

a!=(c==d)

等。

关系表达式的值是真”和“假"，用“1"和"0"表示。

如：

5>0 的值为"真"，即为 1。

(a=3)>(b=5)由于 3>5 不成立，故其值为假，即为 0。

【例 5.1】

```
main(){
  char c='k';
  int i=1,j=2,k=3;
  float x=3e+5,y=0.85;
  printf("%d,%d\n",'a'+5<c,-i-2*j>=k+1);
  printf("%d,%d\n",1<j<5,x-5.25<=x+y);
  printf("%d,%d\n",i+j+k==-2*j,k==j==i+5);
 }
```

在本例中求出了各种关系运算符的值。字符变量是以它对应的 ASCII 码参与运算的。对于含多个关系运算符的表达式，如 k==j==i+5,根据运算符的左结合性，先计算 k==j,该式不成立，其值为 0，再计算 0==i+5，也不成立，故表达式值为 0。

1.1 逻辑运算符和表达式

1.1.1 逻辑运算符极其优先次序

C 语言中提供了三种逻辑运算符：

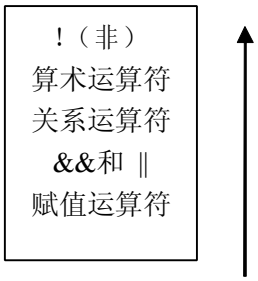
1)　&&　与运算

2)　||　或运算

3)　!　非运算

与运算符&&和或运算符||均为双目运算符。具有左结合性。非运算符!为单目运算符，具有右结合性。逻辑运算符和其它运算符优先级的关系可表示如下：

!（非）→&&(与)→||(或)



“&&”和“||”低于关系运算符，“！”高于算术运算符。

按照运算符的优先顺序可以得出：

```
 a>b && c>d     等价于     (a>b)&&(c>d)
!b==c||d<a      等价于     ((!b)==c)||(d<a)
a+b>c&&x+y<b    等价于     ((a+b)>c)&&((x+y)<b)
```

1.1.1 逻辑运算的值

逻辑运算的值也为“真”和“假”两种，用“1”和“0 ”来表示。其求值规则如下：

1. 与运算 &&：参与运算的两个量都为真时，结果才为真，否则为假。

   例如：

   5>0 && 4>2

   由于 5>0 为真，4>2 也为真，相与的结果也为真。

2. 或运算||：参与运算的两个量只要有一个为真，结果就为真。 两个量都为假时，结果为假。

   例如：

   5>0||5>8

   由于 5>0 为真，相或的结果也就为真。

3. 非运算!：参与运算量为真时，结果为假；参与运算量为假时，结果为真。

   例如：

   !(5>0)

   的结果为假。

虽然 C 编译在给出逻辑运算值时，以“1”代表“真”，“0 ”代表“假”。 但反过来在判断一个量是为“真”还是为“假”时，以“0”代表“假”，以非“0”的数值作为“真”。例如：

由于 5 和 3 均为非“0”因此 5&&3 的值为“真”，即为 1。

又如：

5||0 的值为“真”，即为 1。

1.1.1 逻辑表达式

逻辑表达式的一般形式为：

表达式　逻辑运算符　表达式

其中的表达式可以又是逻辑表达式，从而组成了嵌套的情形。

例如：

(a&&b)&&c

根据逻辑运算符的左结合性，上式也可写为：

a&&b&&c

逻辑表达式的值是式中各种逻辑运算的最后值，以“1”和“0”分别代表“真”和“假”。

【例 5.2】

```
main(){
    char c='k';
    int i=1,j=2,k=3;
    float x=3e+5,y=0.85;
    printf("%d,%d\n",!x*!y,!!!x);
    printf("%d,%d\n",x||i&&j-3,i<j&&x<y);
    printf("%d,%d\n",i==5&&c&&(j=8),x+y||i+j+k);
}
```

本例中!x 和!y 分别为 0，!x*!y 也为 0，故其输出值为 0。由于 x 为非 0，故!!!x 的逻辑值为 0。对 x|| i && j-3 式，先计算 j-3 的值为非 0，再求 i && j-3 的逻辑值为 1，故 x||i&&j-3 的逻辑值为 1。对 i<j&&x<y 式，由于 i<j 的值为 1，而 x<y 为 0 故表达式的值为 1，0 相与，最后为 0，对 i==5&&c&&(j=8)式，由于 i==5 为假，即值为 0，该表达式由两个与运算组成，所以整个表达式的值为 0。对于式 x+ y||i+j+k 由于 x+y 的值为非 0，故整个或表达式的值为 1。

1.1 if 语句

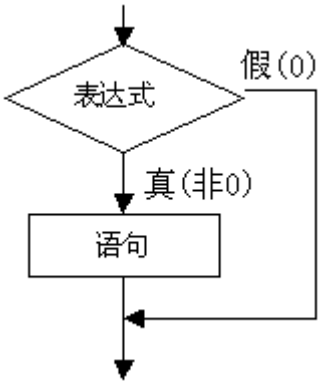
用 if 语句可以构成分支结构。它根据给定的条件进行判断，以决定执行某个分支程序段。C 语言的 if 语句有三种基本形式。

1.1.1 if 语句的三种形式

1. 第一种形式为基本形式：if

if(表达式) 语句

其语义是：如果表达式的值为真，则执行其后的语句， 否则不执行该语句。其过程可表示为下图。



【例 5.3】

```
main(){
    int a,b,max;
    printf("\n input two numbers:   ");
    scanf("%d%d",&a,&b);
    max=a;
    if (max<b) max=b;
    printf("max=%d",max);
}
```

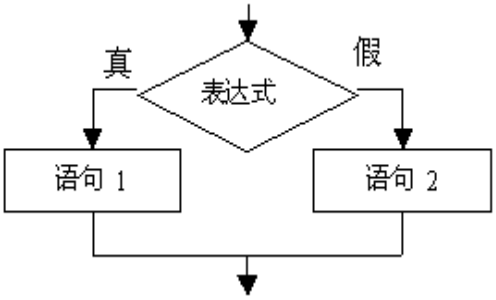
本例程序中，输入两个数 a,b。把 a 先赋予变量 max，再用 if 语句判别 max 和 b 的大小，如 max 小于 b，则把 b 赋予 max。因此 max 中总是大数，最后输出 max 的值。

2. 第二种形式为：if-else

```
if(表达式)
  语句 1;
else
  语句 2;
```

其语义是：如果表达式的值为真，则执行语句 1，否则执行语句 2 。

其执行过程可表示为下图。



【例 5.4】

```
main(){
    int a, b;
    printf("input two numbers:     ");
    scanf("%d%d",&a,&b);
    if(a>b)
      printf("max=%d\n",a);
    else
      printf("max=%d\n",b);
}
```

输入两个整数，输出其中的大数。

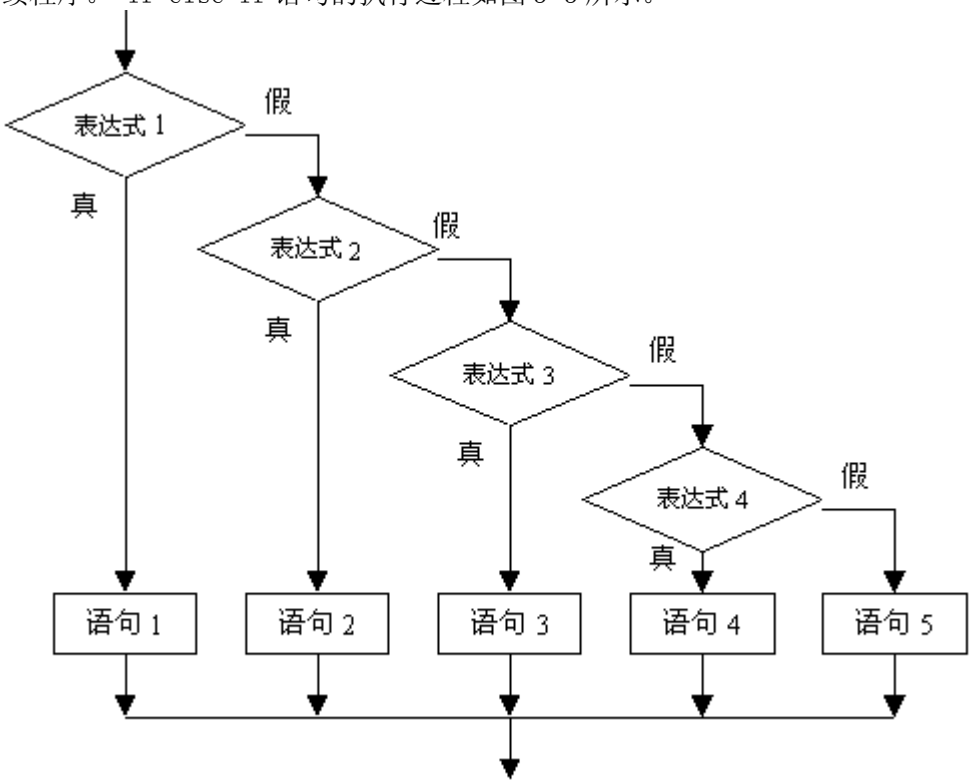
改用 if-else 语句判别 a,b 的大小，若 a 大，则输出 a，否则输出 b。

3. 第三种形式为 if-else-if 形式

前二种形式的 if 语句一般都用于两个分支的情况。 当有多个分支选择时，可采用 if-else-if 语句，其一般形式为：

```
if(表达式 1)
    语句 1;
else  if(表达式 2)
    语句 2;
else  if(表达式 3)
    语句 3;
    …
else  if(表达式 m)
    语句 m;
else
    语句 n;
```

其语义是：依次判断表达式的值，当出现某个值为真时，则执行其对应的语句。然后跳到整个 if 语句之外继续执行程序。 如果所有的表达式均为假，则执行语句 n。然后继续执行后续程序。 if-else-if 语句的执行过程如图 3–3 所示。



【例 5.5】

```
#include"stdio.h"
main(){
    char c;
    printf("input a character:    ");
    c=getchar();
    if(c<32)
      printf("This is a control character\n");
    else if(c>='0'&&c<='9')
```

```
    printf("This is a digit\n");
  else if(c>='A'&&c<='Z')
    printf("This is a capital letter\n");
  else if(c>='a'&&c<='z')
    printf("This is a small letter\n");
  else
    printf("This is an other character\n");
}
```

本例要求判别键盘输入字符的类别。可以根据输入字符的 ASCII 码来判别类型。由 ASCII 码表可知 ASCII 值小于 32 的为控制字符。 在"0"和"9"之间的为数字，在"A"和"Z"之间为大写字母， 在"a"和"z"之间为小写字母，其余则为其它字符。 这是一个多分支选择的问题，用 if-else-if 语句编程，判断输入字符 ASCII 码所在的范围，分别给出不同的输出。例如输入为"g"，输出显示它为小写字符。

4. 在使用 if 语句中还应注意以下问题:

1) 在三种形式的 if 语句中，在 if 关键字之后均为表达式。 该表达式通常是逻辑表达式或关系表达式， 但也可以是其它表达式，如赋值表达式等，甚至也可以是一个变量。

例如：

```
if(a=5) 语句;
if(b) 语句;
```

都是允许的。只要表达式的值为非 0，即为"真"。

如在:

```
if(a=5)…;
```

中表达式的值永远为非 0，所以其后的语句总是要执行的，当然这种情况在程序中不一定会出现，但在语法上是合法的。

又如，有程序段：

```
if(a=b)
   printf("%d",a);
else
   printf("a=0");
```

本语句的语义是，把 b 值赋予 a，如为非 0 则输出该值，否则输出"a=0"字符串。这种用法在程序中是经常出现的。

2) 在 if 语句中，条件判断表达式必须用括号括起来，在语句之后必须加分号。

3) 在 if 语句的三种形式中，所有的语句应为单个语句，如果要想在满足条件时执行一组(多个)语句，则必须把这一组语句用{}括起来组成一个复合语句。但要注意的是在}之后不能再加分号。

例如：

```
if(a>b)
  {a++;
   b++;}
else
   {a=0;
    b=10;}
```

1.1.1 if 语句的嵌套

当 if 语句中的执行语句又是 if 语句时，则构成了 if 语句嵌套的情形。

其一般形式可表示如下：

```
if(表达式)
if 语句;
```

或者为

```
if(表达式)
   if 语句;
else
   if 语句;
```

在嵌套内的 if 语句可能又是 if-else 型的，这将会出现多个 if 和多个 else 重叠的情况，这时要特别注意 if 和 else 的配对问题。

例如：

```
if(表达式 1)
if(表达式 2)
    语句 1;
else
    语句 2;
```

其中的 else 究竟是与哪一个 if 配对呢?

应该理解为：

```
 if(表达式 1)
   if(表达式 2)
      语句 1;
   else
      语句 2;
```

还是应理解为：

```
if(表达式 1)
   if(表达式 2)
      语句 1;
else
   语句 2;
```

为了避免这种二义性，C 语言规定，else 总是与它前面最近的 if 配对，因此对上述例子应按前一种情况理解。

【例 5.6】

```
main(){
    int a,b;
    printf("please input A,B:    ");
    scanf("%d%d",&a,&b);
    if(a!=b)
    if(a>b)  printf("A>B\n");
    else     printf("A<B\n");
    else     printf("A=B\n");
```

```
}
```

比较两个数的大小关系。

本例中用了 if 语句的嵌套结构。采用嵌套结构实质上是为了进行多分支选择，实际上有三种选择即 A>B、A<B 或 A=B。这种问题用 if-else-if 语句也可以完成。而且程序更加清晰。因此，在一般情况下较少使用 if 语句的嵌套结构。以使程序更便于阅读理解。

【例 5.7】

```
main(){
    int a,b;
    printf("please input A,B:       ");
    scanf("%d%d",&a,&b);
    if(a==b) printf("A=B\n");
    else if(a>b)  printf("A>B\n");
    else  printf("A<B\n");
}
```

1.1.1 条件运算符和条件表达式

如果在条件语句中，只执行单个的赋值语句时， 常可使用条件表达式来实现。不但使程序简洁，也提高了运行效率。

条件运算符为?和:，它是一个三目运算符，即有三个参与运算的量。

由条件运算符组成条件表达式的一般形式为：

表达式 1?　表达式 2：　表达式 3

其求值规则为：如果表达式 1 的值为真，则以表达式 2 的值作为条件表达式的值，否则以表达式 2 的值作为整个条件表达式的值。

条件表达式通常用于赋值语句之中。

例如条件语句：

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

可用条件表达式写为

```
max=(a>b)?a:b;
```

执行该语句的语义是：如 a>b 为真，则把 a 赋予 max，否则把 b 赋予 max。

使用条件表达式时，还应注意以下几点：

1) 条件运算符的运算优先级低于关系运算符和算术运算符，但高于赋值符。

因此

```
max=(a>b)?a:b
```

可以去掉括号而写为

```
max=a>b?a:b
```

2) 条件运算符?和：是一对运算符，不能分开单独使用。

3) 条件运算符的结合方向是自右至左。

例如：

a>b?a:c>d?c:d

应理解为

a>b?a:(c>d?c:d)

这也就是条件表达式嵌套的情形，即其中的表达式 3 又是一个条件表达式。

【例 5.8】

```
main(){
    int a,b,max;
    printf("\n input two numbers:    ");
    scanf("%d%d",&a,&b);
    printf("max=%d",a>b?a:b);
}
```

用条件表达式对上例重新编程，输出两个数中的大数。

1.1 switch 语句

C 语言还提供了另一种用于多分支选择的 switch 语句， 其一般形式为：

```
switch(表达式){
case 常量表达式 1:  语句 1;
case 常量表达式 2:  语句 2;
…
case 常量表达式 n:  语句 n;
default        :  语句 n+1;
}
```

其语义是：计算表达式的值。 并逐个与其后的常量表达式值相比较，当表达式的值与某个常量表达式的值相等时， 即执行其后的语句，然后不再进行判断，继续执行后面所有 case 后的语句。如表达式的值与所有 case 后的常量表达式均不相同时，则执行 default 后的语句。

【例 4.9】

```
main(){
    int a;
    printf("input integer number:      ");
    scanf("%d",&a);
    switch (a){
    case 1:printf("Monday\n");
    case 2:printf("Tuesday\n");
    case 3:printf("Wednesday\n");
    case 4:printf("Thursday\n");
    case 5:printf("Friday\n");
    case 6:printf("Saturday\n");
    case 7:printf("Sunday\n");
    default:printf("error\n");
    }
```

```
}
```

本程序是要求输入一个数字，输出一个英文单词。但是当输入 3 之后，却执行了 case3 以及以后的所有语句，输出了 Wednesday 及以后的所有单词。这当然是不希望的。为什么会出现这种情况呢?这恰恰反应了 switch 语句的一个特点。在 switch 语句中，"case 常量表达式"只相当于一个语句标号， 表达式的值和某标号相等则转向该标号执行，但不能在执行完该标号的语句后自动跳出整个 switch 语句，所以出现了继续执行所有后面 case 语句的情况。 这是与前面介绍的 if 语句完全不同的，应特别注意。为了避免上述情况，C 语言还提供了一种 break 语句，专用于跳出 switch 语句，break 语句只有关键字 break，没有参数。在后面还将详细介绍。修改例题的程序，在每一 case 语句之后增加 break 语句，使每一次执行之后均可跳出 switch 语句，从而避免输出不应有的结果。

【例 4.10】

```
main(){
    int a;
    printf("input integer number:    ");
    scanf("%d",&a);
    switch (a){
      case 1:printf("Monday\n");break;
      case 2:printf("Tuesday\n"); break;
      case 3:printf("Wednesday\n");break;
      case 4:printf("Thursday\n");break;
      case 5:printf("Friday\n");break;
      case 6:printf("Saturday\n");break;
      case 7:printf("Sunday\n");break;
      default:printf("error\n");
    }
}
```

在使用 switch 语句时还应注意以下几点：

1) 在 case 后的各常量表达式的值不能相同，否则会出现错误。
2) 在 case 后，允许有多个语句，可以不用{}括起来。
3) 各 case 和 default 子句的先后顺序可以变动，而不会影响程序执行结果。
4) default 子句可以省略不用。

1.1 程序举例

【例 4.11】输入三个整数，输出最大数和最小数。

```
main(){
    int a,b,c,max,min;
    printf("input three numbers:    ");
    scanf("%d%d%d",&a,&b,&c);
    if(a>b)
```

```
    {max=a;min=b;}
  else
    {max=b;min=a;}
  if(max<c)
   max=c;
  else
    if(min>c)
  min=c;
  printf("max=%d\nmin=%d",max,min);
}
```

本程序中，首先比较输入的 a,b 的大小，并把大数装入 max，小数装入 min 中，然后再与 c 比较，若 max 小于 c，则把 c 赋予 max；如果 c 小于 min，则把 c 赋予 min。因此 max 内总是最大数，而 min 内总是最小数。最后输出 max 和 min 的值即可。

【例 4.12】计算器程序。用户输入运算数和四则运算符，输出计算结果。

```
main(){
  float a,b;
  char c;
  printf("input expression: a+(-,*,/)b \n");
  scanf("%f%c%f",&a,&c,&b);
  switch(c){
     case '+': printf("%f\n",a+b);break;
     case '-': printf("%f\n",a-b);break;
     case '*': printf("%f\n",a*b);break;
     case '/': printf("%f\n",a/b);break;
     default: printf("input error\n");
  }
}
```

本例可用于四则运算求值。switch 语句用于判断运算符，然后输出运算值。当输入运算符不是+,-,*,/时给出错误提示。