计算机程序设计基础(1)
--- C语言程序设计(5)

孙甲松
sunjiasong@tsinghua.edu.cn
电子工程系 信息认知与智能系统研究所
罗姆楼6-104
电话: 13901216180/62796193
2022.10.

第5章 选择结构

5.1 语句与复合语句

C程序以语句为基本单位

终结符

C程序

简单语句

- 表达式语句是一个表达式后面跟随一个分号(;)所构成的语句。
- 空语句(;)中只包括一个分号，实现空操作。
- 流程控制语句(如 **break; continue;**等)。
- 函数返回语句 **return x;** 中的分号前不是表达式，只是实现某种控制操作，但它们也都是以分号结束。

复合语句

- 在一个函数体内部，由左、右花括号括起来的语句称为复合语句，它的一般形式为:

```
{  说明部分;
   语句部分;
}
```

由此可以看出，在C程序中，一个函数的函数体实际上就是一个复合语句。

下面对于复合语句作几点说明：

- 一个复合语句在语法上等同于一个独立的语句，因此，在程序中，凡是单个语句(如表达式语句)能够出现的地方都可以出现复合语句，并且，复合语句作为一个语句又可以出现在其他复合语句的内部。 如【例5-1】。

- 复合语句是以右花括号为结束标志，因此，在复合语句右括号的后面不必加分号，但在复合语句内的最后一个非复合语句须以分号作为结束符。

【例5-1】设有下列C程序：

```
#include  <stdio.h>
main( )
{     int  y;
      y=100;
      {       int x;   /* 局部变量 */
              x=20;
              {    int  a; /* 局部变量 */
                   a=y;
                   printf("a=%d\n", a);
                   printf("x=%d\n", x);
              }
      }
      printf("y=%d\n", y);
}
```

复合语句

- 复合语句是可以嵌套的

- 在复合语句的嵌套结构（将函数体也看成是一个复合语句，而且是最外层的复合语句）中，一个复合语句内所进行的说明只适合于本层中该说明语句以后的部分（包括其内层的复合语句），在该复合语句外不起作用。

【例5-2】 设有下列C程序：

```
#include <stdio.h>
main( )
{   int  y;
    y=100;
    {   int x;
        x=20;
        printf("x=%d\n", x);
    }
    printf("y=%d\n", y);
    printf("x=%d\n", x);
}
```

复合语句

变量的作用域与生命周期

在编译时会出现如下错误：

05-02.c(10) : error C2065: “x”: 未声明的标识符

第10行 printf ("x=%d\n", x); 所打印的x是未定义的标识符，因为前面复合语句中定义的x已经不存在了。

●在复合语句的嵌套结构中，如果在内层与外层定义了同名的变量，则按照局部优先的原则，内层复合语句中的变量掩蔽外层复合语句的同名变量，直到内层复合语句结束，外层复合语句的同名变量才可以访问到。而且内层复合语句中对内层定义的变量的执行结果也不带回到外层。

【例5-3】 设有下列程序：

```
#include <stdio.h>
main( )
{       int  x, y;
        x=10;
        y=100;
        {   x=20;
            printf("y=%d\n", y);
            printf("x=%d\n", x);
        }
        printf("x=%d\n", x);
}
```

```
{  int x;
   x=20;
   printf("y=%d\n", y);
   printf("x=%d\n",  x);
}
```

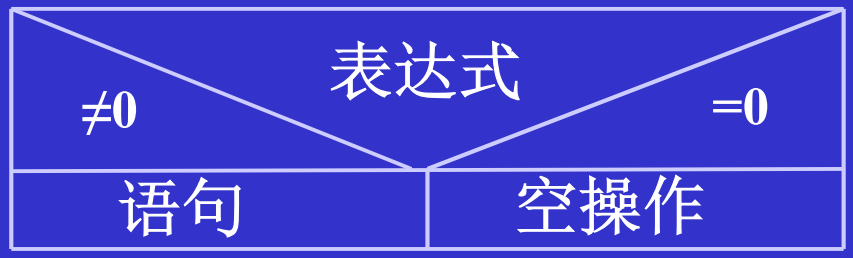
```
y=100
x=20
x=20
```

```
y=100
x=20
x=10
```

变量的掩蔽现象

5.2 if 语句

- if 语句的形式为：if (表达式) 语句



if 语句的功能是：

若表达式值为1(或非0)，则执行表达式后面的语句，执行完该语句后继续执行if语句后的语句；

若表达式值为0，则不执行表达式后面的语句而直接执行if 语句后的语句。如果“表达式”后面的是复合语句，则要用一对花括号{ }括起来。这种选择结构的NS流程图如上图所示。

【例5-4】 计算并输出下列分段函数值：

$$Y=\begin{cases}-2 & x<0\\ 2 & x\geq 0\end{cases}$$

```
#include   <stdio.h>
main( )
{   double  x, y;
    printf("input x: ");
    scanf("%lf", &x);
    y=-2;
    if (x>=0) y=2;
    printf("y=%f\n", y);
}
```

例如，if语句：

if (a !=0) printf("%d\n", a);

的功能是：如果变量a的值不等于0，则打印输出变量a的值。

又如，if语句

if (a>b) { t=a; a=b; b=t; }

的功能是：

如果变量a的值大于变量b的值，则将变量a与b的值交换。

在C语言程序中，常会看到：

if (a) printf("%d\n", a);

语义是：若a不等于0，则打印a的值。它等价于：

if (a != 0) printf("%d\n", a);

同样：　**if (!a) printf("%d\n", a);**

等价于：　**if (a == 0) printf("%d\n", a);**

- if 语句中的逻辑表达式(即条件)必须要用一对圆括号括起来

例如，如果将下面的if语句

```
if (x>=0) y=2;
```

写成

```
if  x>=0  y=2;
```

就会出现编译错误。

- if 语句后的语句可以是复合语句

例如，下列if语句是合法的：

```
x=3; y=4;
if (a>b) { x=1; y=2; }
   …
```

切记不能在{ }前加分号：

```
x=3; y=4;
if (a>b) ; { x=1; y=2; }
```

这样写使得 { } 与if无关。

在这个程序段中，首先为变量x与y赋值3与4，然后判断变量a与b的大小，若a的值大于b的值，则将变量x与y的值修改为1与2。但要注意，if语句的作用范围与它后面语句出现的位置无关。

例如，下列if语句是合法的：

```
if (a>b) x=1; y=2;
```

千万不要误认为只有当a>b为真时，才执行x=1; y=2;， 其实y=2; 虽然写在if条件的后面，根本不受if条件约束，无论a>b值为真还是假，都会被执行。if条件只约束x=1; 当a>b为真时，才执行x=1;语句，否则x=1;语句不会被执行。

- 在使用if语句时，一定要注意逻辑表达式的正确写法，特别是在进行数值型数据比较时，一定要注意小心使用等于比较运算符的使用。由于计算机中的实数一般都是近似的，对实数进行不同的运算过程其结果可能是不同的，它们之间有一定的误差。因此，对于理论上应该相等的两个实数，在用等于运算符“==”进行比较时，等到的结果可能是不相等的。

【例5-5】 下面C程序的功能是用两个不同的计算过程计算10个实数0.1进行累加，并比较它们的计算结果：

```
#include  <stdio.h>
main( )
{ int  k, flag;
  double  x, y;
  x=1.0;   /*  用赋值语句直接赋值为1.0 */
  y=0.0;   /* 用10个赋值语句逐步累加 */
  y=y+0.1; y=y+0.1; y=y+0.1; y=y+0.1; y=y+0.1;
  y=y+0.1; y=y+0.1; y=y+0.1; y=y+0.1; y=y+0.1;
  flag=0;
  if (x == y)  flag=1;      /*如果x与y值相等则置flag的值为1*/
  printf("flag=%d\n", flag);
}
```

```
#include  <math.h>
```

```
printf("x=%20.17f\n", x);
printf("y=%20.17f\n", y);
```

```
if (fabs(x-y)<1.0e-10)
```

上面程序运行结果为：

flag=0

- 显然，这个输出结果与理论上的结果不符合。理论上x与y的值应该是相同的，即表达式x == y的值应为1(条件满足)，通过执行if语句

```
if (x == y)  flag=1;
```

后，整型变量flag的值应变为1。但实际输出的flag值却为0，这说明计算得到的结果与预期值是不一样的，实际的x与y值并不相同。

- 如果在程序中增加两个输出x与y值的语句，输出结果为：

x= 1.0000000000000000

y= 0.9999999999999989

flag=0

- 由输出结果可以看出，果然x与y的值不同，这是因为0.1在计算机中的表示是近似的，导致计算结果可能与实际不符。并且，不同的计算过程所得到的结果可能是不同的。因此，不能直接判断两个浮点数计算结果是否相等（即两者相减后的结果是否等于0），而应该利用一个参考值，当它们相减后的绝对值小于这个参考值时，就认为它们相等。例如， **fabs(x-y)<1.0e-10**

- 用多个if 语句也可以实现多路分支选择结构，但要特别注意条件表达式的正确写法。

【例5-6】从键盘读入一个成绩，如果成绩在85～100分之间，则输出"Excellent!"；如果成绩在70～84分之间，则输出"Good!"；如果成绩在60～69分之间，则输出"Pass!"；如果成绩在60分以下，则输出"No pass!"。

多路分支选择结构

```
#include   <stdio.h>
 main( )
 {  float  grade;
    printf("input grade : ");
    scanf("%f", &grade);
    if  (grade>=85.0)  printf("Excellent!\n");
    if  (grade >=70.0 && grade <85.0)  printf("Good!\n");
    if  (grade >=60.0 && grade <70.0)  printf("Pass!\n");
    if  (grade <60.0)  printf("No pass!\n");
 }
```

如果【例5-6】改为：

```
#include   <stdio.h>
main( )
{  float  grade;
   printf("input grade : ");
   scanf("%f", &grade);
   if  (grade>=85.0)   printf("Excellent!\n");
   if  (grade >=70.0)  printf("Good!\n");
   if  (grade >=60.0)  printf("Pass!\n");
   if  (grade <60.0)    printf("No pass!\n");
}
```

运行：

```
input grade: 90
Excellent!
Good!
Pass!
```

结果是错的！

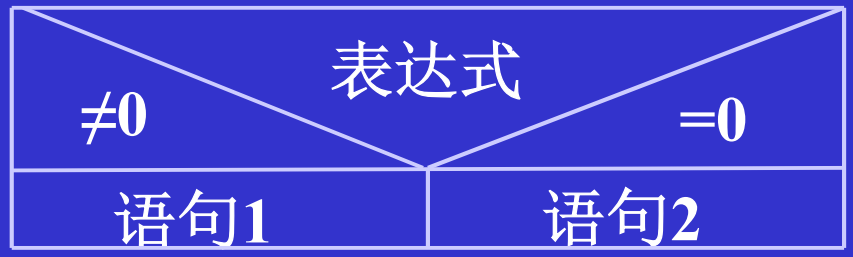
原因是：几个if语句之间的取值范围不是互斥的！

5.3. if…else结构

- 语句形式为　if (表达式) 语句1
  else 语句2

else语句本身不能单独存在

功能 若表达式值为1(非0)，则执行语句1，否则执行语句2。其中语句1与语句2均可以是复合语句。



由此图可以看出，if…else结构可以实现两路分支选择结构。C语言允许if…else结构的嵌套。即在if…else结构中，语句1与语句2中又可以包含完整的if语句或if…else结构，并且，这种嵌套可以多层。利用if…else结构的嵌套，可以实现多路分支选择结构。

如果【例5-6】改为：

```
#include   <stdio.h>
main( )
{  float  grade;
   printf("input grade : ");
   scanf("%f", &grade);
   if  (grade>=85.0)  printf("Excellent!\n");
   else if  (grade >=70.0)  printf("Good!\n");
   else if  (grade >=60.0)  printf("Pass!\n");
   else if  (grade <60.0)  printf("No pass!\n");
}
```

运行：

```
input grade: 90
Excellent!
```

利用else使得几个if语句之间的取值范围是互斥的。

最后一行的if (grade <60.0) 可以不写。

【例5-7】计算并输出下列分段函数值：

$$Y=\begin{cases}0 & x<-10;\\ 2x+20 & -10\le x<0;\\ 20 & 0\le x<20;\\ 30-0.5x & 20\le x<40;\\ 50-x & 40\le x<50;\\ 0 & x\ge 50;\end{cases}$$

其中x从键盘输入。

如果输入的x值大于等于50，只需要判断1次

但如果输入的x值小于-10，则需要判断5次

```
#include   <stdio.h>
main( )
{ float x, y;
  printf("input x: ");
  scanf("%f", &x);
  if (x>=50.0) y=0.0;
  else  if (x>=40.0) y=50-x;
    else  if (x>=20.0) y=30-0.5*x;
      else  if (x>=0.0) y=20.0;
        else  if (x>=-10.0) y=2*x+20;
          else  y=0.0;
  printf("x=%f, y=%f\n", x, y);
}
```

```
if (x<-10.0) y=0.0;
  else if (x<0.0) y=2*x+20;
   else if (x<20.0) y=20.0;
    else if (x<40.0) y=30-0.5*x;
     else if (x<50.0) y=50-x;
      else y=0.0;
```

x值大于50也需要判断5次

提醒：在处理多路分支选择时，应尽量将出现几率高的条件写在前面，以提高程序的执行效率。

改变一下【例5-7】程序的判断顺序：

```
#include   <stdio.h>
main( )
{ float x, y;
  printf("input x: ");
  scanf("%f", &x);
  if (x>=20.0)
  {  if (x>=50.0) y=0.0;
     else  if (x>=40.0) y=50-x;
           else  y=30-0.5*x;
  }
  else
  {  if (x>=0.0) y=20.0;
     else  if (x>=-10.0) y=2*x+20;
           else  y=0.0;
  }
  printf("x=%f, y=%f\n", x, y);
}
```

无论输入的x值是什么，最多需要判断3次即可得出结果！

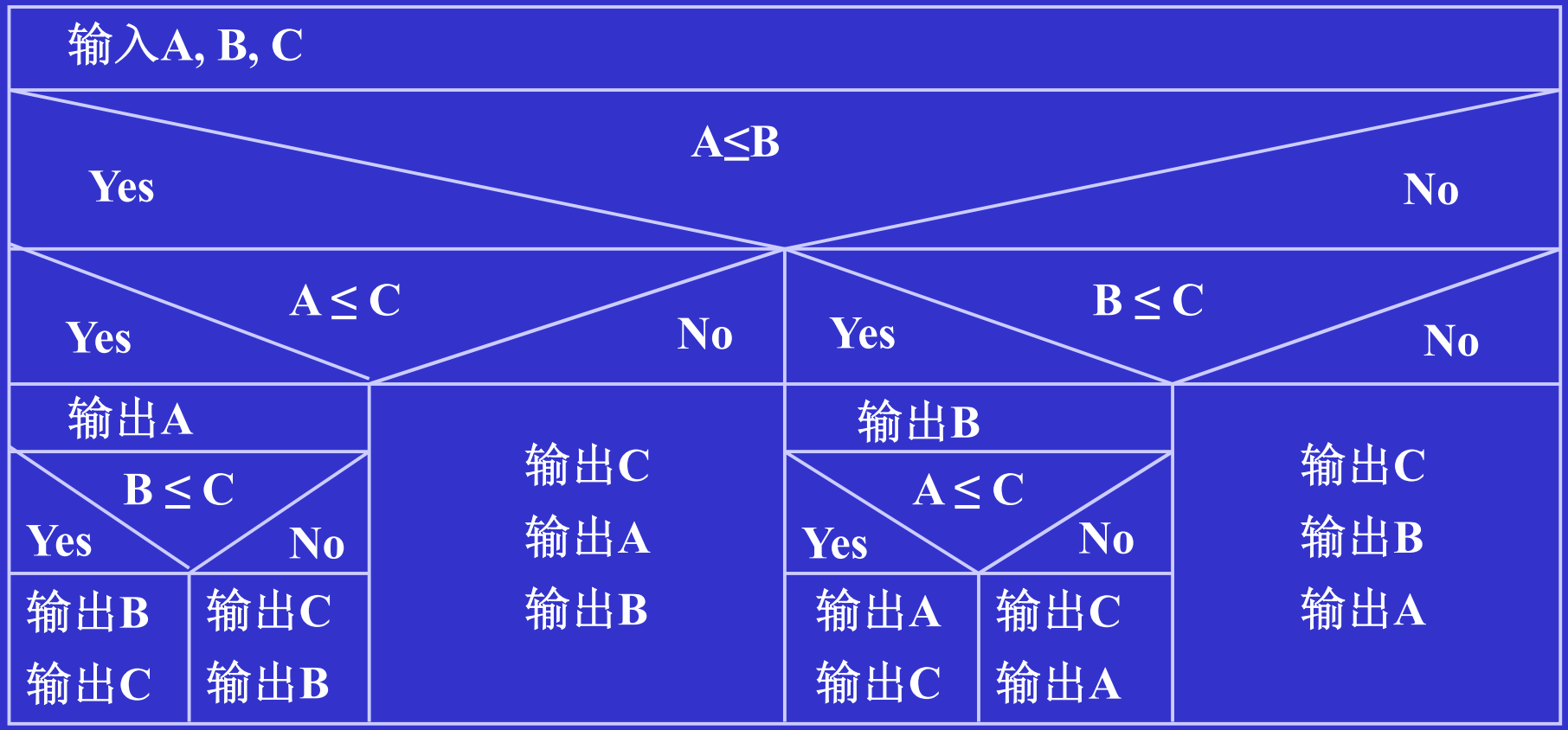
再次提醒：不能在else和随后的语句或复合语句{ }之间加分号，例如：

```
if (x>=-10.0) y=2*x+20;
else ;  y=0.0;
```

这样写使得y=0.0;与if无关。

提醒：多运用程序设计的技巧，以提高程序的执行效率。

【例5-8】由键盘输入三个整数A、B、C，然后按从小到大的顺序输出。



```
#include   <stdio.h>
#define PR(x)  printf("%d\n", x)
main( )
{ int a, b, c;
  printf("input a, b, c: ");
  scanf("%d%d%d", &a, &b, &c);
  if (a<=b)
  {     if (a<=c)
        { PR(a);
          if (b<=c)  { PR(b);  PR(c); }
          else {  PR(c);  PR(b); }
        }
```

先判断A与B的值。

①如果A≤B，则再判断A与C的值:

若A≤C，则A为三个数中最小者，输出A；接着判断B与C的值:

若B≤C，则依次输出B与C，否则依次输出C与B。

若A>C，则依次输出C、A与B。

```
        else  /* a>c */
        {   PR(c); PR(a); PR(b); }
    }
    else  /* a>b */
    {   if (b<=c)
        {   PR(b);
            if (a<=c) { PR (a); PR(c); }
            else { PR(c); PR(a); }
        }
        else /* b>c */
        {  PR(c); PR(b); PR(a); }
    }
}
```

② 如果A>B, 则再判断B与C的值:

若B≤C，则B为三个数中最小者，输出B；接着判断A与C的值:

若A≤C，则依次输出A与C，否则依次输出C与A。

若B>C，则依次输出C、B与A。

对if…else结构的几点说明

- if…else结构中的语句1与语句2都可以是复合语句。
- 在if…else结构中，语句1与语句2都可以是空语句。

下列if…else结构是合法的：
```
if (a>b) {x=1; y=2; }
else ;
```
它等价于下列if语句：
```
if (a>b) {x=1; y=2; }
```

又如，下列if…else结构也是合法的：
```
if (a>b) ;
else {x=3; y=4; }
```
它等价于下列if语句：
```
if (a<=b) {x=3; y=4; }
```
注意，在这个if语句中的条件刚好与原来的条件相反。

再如，下列if…else结构也是合法的：
```
if (a>b) ;
else ;
```
它等价于一个空操作语句。

- 如果在 else 前面有多个 if 语句，则else与同层最近的if 配对，与书写方式无关！

【例5-9】 设有下列C程序：

```
#include   <stdio.h>
main( )
{ int  x, y;
  scanf("%d", &x);
  y=-1;
  if  (x!=0)
    if  (x>0)  y=1;
  else    y=0;
  printf("y=%d\n", y);
}
```

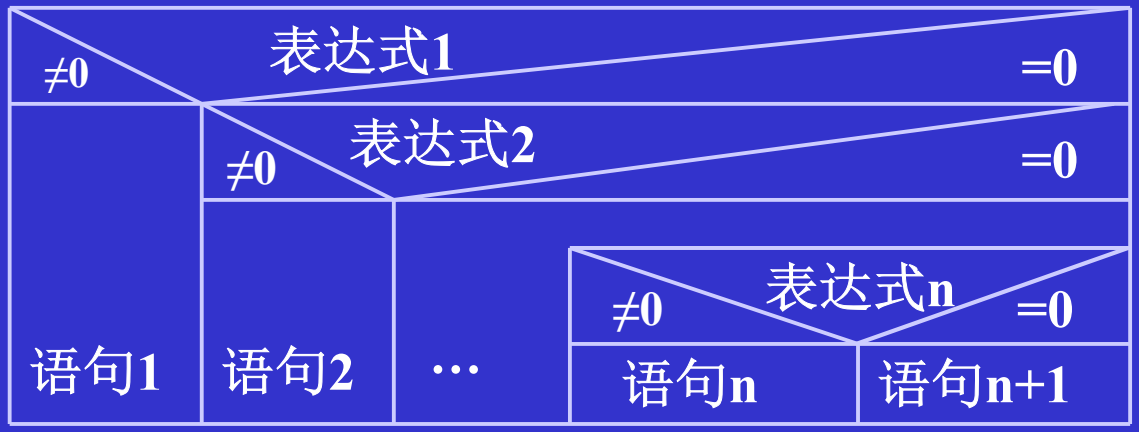
在运行上述程序时，如果从键盘输入-1，则运行结果为 y=0

{ if (x>0) y=1; }

当输入-1时，输出为y=-1。

- 如果有多个if……else结构嵌套如下：

```
if (表达式1) 语句1
else
    if (表达式2) 语句2
    else
        …
        else
            if (表达式n) 语句n
            else 语句n+1
```



则可简写成：

```
if (表达式1) 语句1
else if (表达式2) 语句2
        ……
else if (表达式n) 语句n
else 语句n+1
```

这种结构又称为if…else if结构

5.4 条件运算符 （C语言唯一的一个三目运算符）

- 条件表达式的一般形式为：表达式1 ? 表达式2 : 表达式3

若表达式1为真，则以表达式2为结果，否则以表达式3为结果。

【例5-10】 从键盘输入一个x，计算并输出下列分段函数值：

$$Y=\begin{cases} x^2-1 & x<0 \\ x^2+1 & x\geq 0 \end{cases}$$

其C程序如下：

```
#include   <stdio.h>
main( )
{ float x, y;
  printf("input x: ");
  scanf("%f", &x);
  y=(x<0) ? (x*x-1) : (x*x+1);
  printf("y=%f\n", y);
}
```

```
#include   <stdio.h>
main( )
{ float x;
  printf("input x: ");
  scanf("%f", &x);
  printf("y=%f\n",
   (x<0) ? (x*x-1) : (x*x+1));
}
```

- 条件运算符的优先级比赋值运算符高。
- 条件运算符的优先级比关系运算符与算术运算符都要低。
- 条件运算符的结合方向为“从右到左”。

【例5-11】 从键盘输入一个字符，如果输入的是英文大写字母，则将它转换成小写字母后输出，否则输出原来输入的字符。

```
#include   <stdio.h>
main( )
{ char ch;
  printf("input ch: ");
  scanf("%c", &ch);
  ch= (ch>= 'A'&& ch<='Z') ? ch- 'A' + 'a' : ch;
  printf("%c\n", ch);
}
```

```
printf("%c\n", (ch>= 'A'&& ch<='Z')?ch-'A'+'a' :ch);
```

例如：求a, b, c中值最大的给d。

用 if 语句：

```
if (a>b)
        if (a>c)   d = a;
        else       d = c;
else
        if (b>c)   d = b;
        else       d = c;
```

或：

```
if (a>b)
{       if (a>c)  d = a;
        else      d = c;
}
else
{       if (b>c)  d = b;
        else      d = c;
}
```

用三目运算符：

```
d = a>b ? (a>c ? a : c) : (b>c? b : c);
```

也可以写为：

```
d = a>b ? a>c ? a : c : b>c? b : c;
```

但显然可读性不好(注意：条件运算符自右至左结合)

5.5 switch语句

【例5-12】

```
#include  <stdio.h>
main( )
{  int  grade;
   printf("input  grade=");
   scanf("%d", &grade);
   switch((int)(grade/10))
   {  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");
   }
}
```

```
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
}
```

- 由switch中表达式的值与常量表达式的匹配程度来决定进入某个case并开始顺序执行随后的语句；或都不匹配执行default（如果有的话）。

NS图：

表达式				
情况1	情况2	…	情况n	其他
语句1	语句2	…	语句n	语句n+1

- switch语句的结构中的表达式、常量表达式1、…、常量表达式n必须是整型或字符型。
- 同一个switch结构中的常量表达式的值必须互不相同，否则就会出现编译错误，因为“表达式”的同一个值对应多种执行方案，这是错误的。
- 在switch结构中，case 与 default 的顺序可以任意，各case之间的顺序也可以任意。
- 在执行switch结构时，当执行完某case的语句后，如果没有遇到终止语句，将顺序继续执行后面case或default的语句，直到全部执行完后面的语句或遇到 break 语句才退出整个switch结构的执行。（switch只负责转入，不负责转出）
- 在switch结构中，如果没有default且“表达式”值不等于任何case 后常量表达式的值，则不做任何事情，直接退出switch结构。

如果在例5.12中，各case后的语句后面都没有break语句，即改成:

```
switch((int)(grade/10))
{     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");
}
```

则当输入成绩100分时，运行结果为:

```
A
B
C
D
E
```

这显然是不对的。

【例5-13】 编制一个C程序，其功能是：首先从键盘依次输入两个实数作为运算对象，然后从键盘再输入一个运算符，最后输出运算结果。其中运算符的符号分别为：加法运算符“+”、减法运算符“-”、乘法运算符“*”或点“.”；除法运算符“/”；在作除法运算时，如果第二个实数为0时，要求输出信息“error!”。如果输入的运算符不是上述所定义的运算符，要求输出信息“Incorrect symbol!”。

```
#include <stdio.h>
main( )
{ double x, y;
  char ch;
  printf("input x and y: "); /*输入两个实数前的提示*/
  scanf("%lf%lf", &x, &y); /*输入两个实数*/
  ch =getchar( ); /*读掉上次输入中最后一个回车符号*/
  printf("input ch: "); /*输入运算符字符前的提示*/
  ch =getchar( ); /*输入运算符字符*/
```

```
    switch(ch)
    { case '+': printf("%f+%f=%f\n", x, y, x+y);
                 break;
      case '-':  printf("%f-%f=%f\n", x, y, x-y);
                 break;
      case '*' :
      case '.' : printf("%f*%f=%f\n", x, y, x*y);
                 break;
      case '/' : if (y==0.0)
                      printf("error!\n"); /*分母为0*/
                 else
                      printf("%f/%f=%f\n", x, y, x/y);
                 break;
      default:  printf("Incorrect symbol!\n");
    }
}
```

输出结果：

input x, y:2.0 3.0
input ch:+
2.000000+3.000000=5.000000

input x, y:2.0 3.0
input ch:-
2.000000-3.000000=-1.000000

input x, y:2.0 3.0
input ch:*
2.000000*3.000000=6.000000

input x, y:2.0 3.0
input ch:.
2.000000*3.000000=6.000000

input x, y:2.0 3.0
input ch:/
2.000000/3.000000=0.666667

input x, y:2.0 0.0
input ch:/
error!

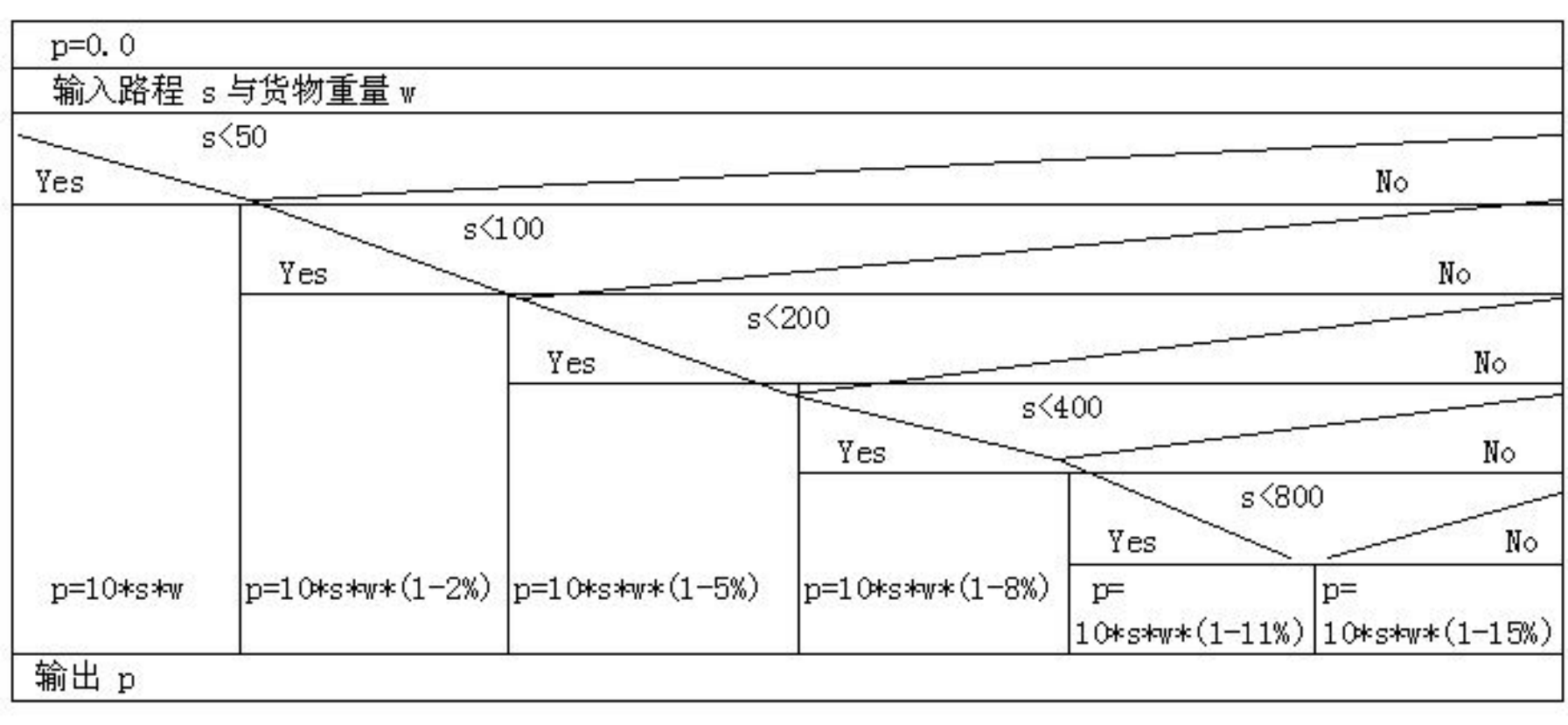
input x, y:2.0 3.0
input ch:,
Incorrect symbol!

【例5-14】 本例可以用if…else if结构实现：

总运费p与路程s、货物重量w的关系可以写成以下分段函数的形式：

$$p=\begin{cases} 10\times s\times w & s<50 \\ 10\times s\times w\times(1-2\%) & 50\le s<100 \\ 10\times s\times w\times(1-5\%) & 100\le s<200 \\ 10\times s\times w\times(1-8\%) & 200\le s<400 \\ 10\times s\times w\times(1-11\%) & 400\le s<800 \\ 10\times s\times w\times(1-15\%) & s\ge 800 \end{cases}$$

用if-else结构处理的NS图：



用if…else if结构的C程序如下：

```
#include  <stdio.h>
main( )
{ double  p, s, w;
  printf("input s=:");
  scanf("%lf", &s);
  printf("input w=:");
  scanf("%lf", &w);
  if (s<=0||w<=0)  p=0.0;
  else if (s<50)  p=10*s*w;
  else if (s<100)  p=10*s*w*(1-0.02);
  else if (s<200)  p=10*s*w*(1-0.05);
  else if (s<400)  p=10*s*w*(1-0.08);
  else if (s<800)  p=10*s*w*(1-0.11);
  else  p=10*s*w* (1-0.15);
  printf("p=%f\n", p);
}
```

本例也可以用switch结构来解决。根据对题意的分析，路程s的变化转折点为50，100，200，400，800，它们都是50的倍数。利用这个特点，可以设置一个整型变量k，其中k的计算公式为

$$k=(int)(s/50)+1$$

数值映射

显然，k的值与路程s之间存在如下关系：

k=1等价于路程s＜50公里；

k=2等价于路程50≤s＜100公里；

k=3，4等价于路程100≤s＜200公里；

5≤k≤8等价于路程200≤s＜400公里；

9≤k≤16等价于路程400≤s＜800公里；

k≥17等价于路程s≥800公里。

上述总运费p与路程s、货物重量w的关系可以写成以下关于k的分段函数形式：

$$p=\begin{cases} 10\times s\times w & k=1 \\ 10\times s\times w\times(1-2\%) & k=2 \\ 10\times s\times w\times(1-5\%) & k=3,4 \\ 10\times s\times w\times(1-8\%) & 5\le k\le 8 \\ 10\times s\times w\times(1-11\%) & 9\le k\le 16 \\ 10\times s\times w\times(1-15\%) & k\ge 17 \end{cases}$$

用switch结构处理的NS图：

p=0.0					
输入路程s与货物重量w					
k=(int)(s/50)+1					
若9≤k≤16 ,置k=9					
若k≥17 ,置k=17					
k					
1	2	3或4	5或6或7或8	9	17
p=10*s*w	p=10*s*w*(1-2%)	p=10*s*w*(1-5%)	p=10*s*w*(1-8%)	p=10*s*w*(1-11%)	p=10*s*w*(1-15%)
输出p					

用switch结构写出C程序如下：

```
#include   <stdio.h>
main( )
{   double p, s, w;   int k;
    printf("input s=: ");
    scanf("%lf", &s);
    printf("input w=: ");
    scanf("%lf", &w);
    if (s<=0||w<=0)  p=0.0;
    else
    {  k=(int)(s/50)+1;
       if (k>=9 && k<=16)  k=9;
       if (k>=17)  k=17;
    switch(k)
    { case 1: p=10*s*w; break;
      case 2: p=10*s*w*(1-0.02); break;
      case 3:
      case 4: p=10*s*w*(1-0.05); break;
      case 5:   case 6:   case 7:
      case 8: p=10*s*w*(1-0.08); break;
      case 9: p=10*s*w*(1-0.11); break;
      case 17: p=10*s*w*(1-0.15);
    }
}
printf("p=%lf\n", p);
}
```

5.6 程序举例

设一元二次方程为 $Ax^2+Bx+C=0$

求解一元二次方程要考虑各种特殊情况，其过程如下：

首先考虑系数A是否等于0。

- 如果A=0, 此时方程变为Bx+C=0，则还要考虑以下两种情况：

  (1)若B=0,则该方程无意义(因为A与B都为0),输出“ERR”,结束;

  (2)若B≠0，则方程只有一个实根，即输出x=-C/B，结束。

- 如果A≠0，需要考虑以下两种情况：

  (1) B=0，此时方程变为 $Ax^2+C=0$ 在这种情况下，

  如果A与C异号，则方程有两个实根为：

$$x_{1,2} = \pm\sqrt{-C/A}$$

如果A与C同号，则方程有两个虚根为：

$$x_{1,2} = \pm j\sqrt{C/A} \quad \text{其中 } j = \sqrt{-1}$$

(2) B≠0，在这种情况下，

如果C=0，则方程变为 $Ax^2+Bx=0$ 两个实根分别为：

$$x_1=0, \quad x_2=-B/A$$

如果C≠0，计算判别式$D=B^2-4AC$，再考虑以下两种情况：

① 如果D≥0，则表示方程有两个实根，公式为：

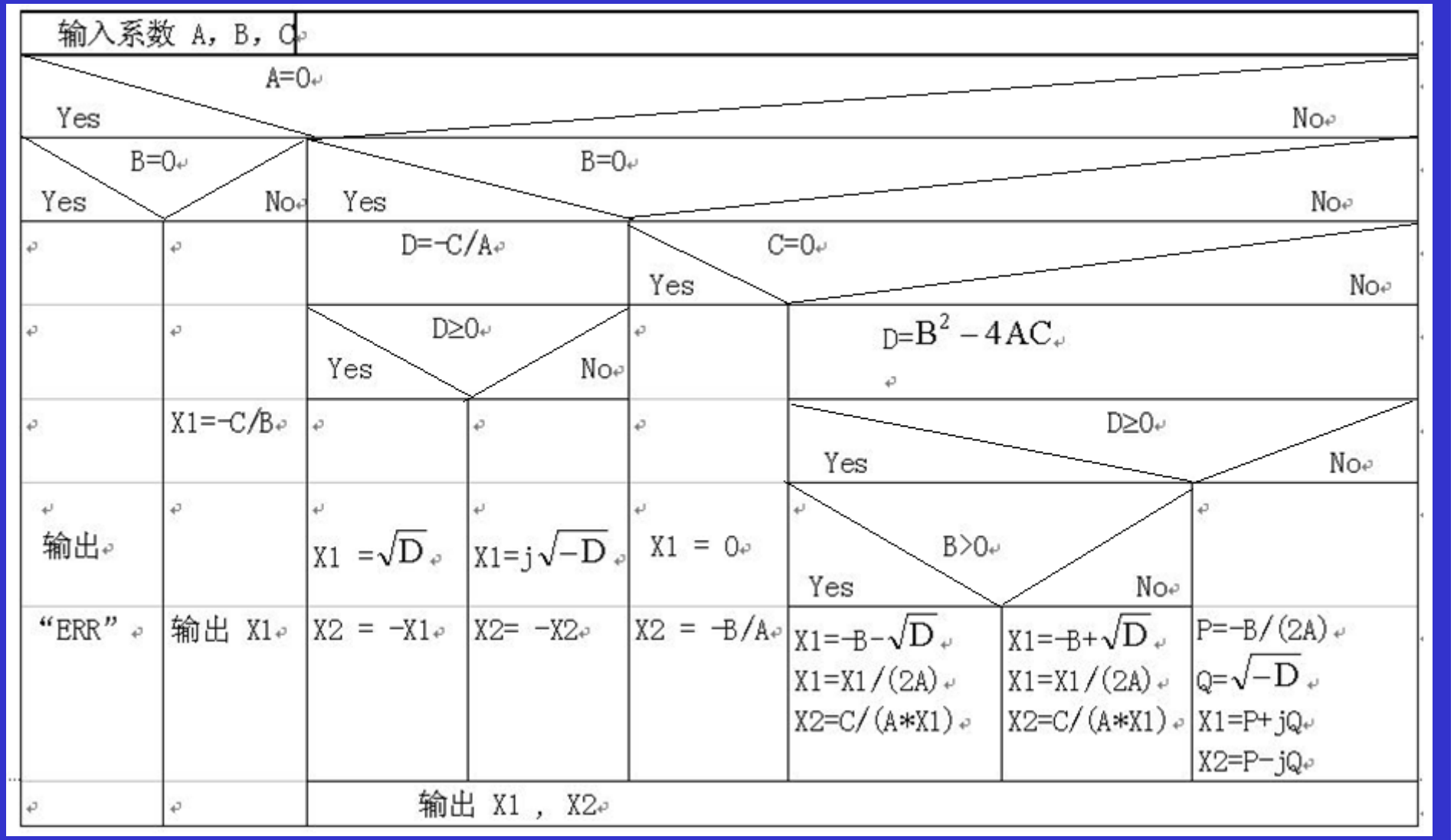
$$x_{1,2} = (-B\pm\sqrt{D})/(2A)$$

② 如果D<0，则表示方程有两个共轭复根，公式为：

$$x_{1,2} = (-B\pm j\sqrt{-D})/(2A)$$

综上所述，可以得到求一元二次方程根的流程图如下图所示。

解一元二次方程流程的NS图：



一元二次方程的求解

```
#include   <stdio.h>
#include  <math.h>
main( )
{  double  a, b, c, d, x1, x2, p;
   printf("input a, b, c: ");
   scanf("%lf%lf%lf", &a, &b, &c);
   if (a==0.0)
   {   if (b==0.0)  printf("ERR\n");
                  /*方程为C=0, 错误*/
       else  printf("X=%f\n", -c/b);
                  /*方程为Bx=C*/
   }
   else if (b==0.0)   /*方程为Ax²+C =0*/
   {      d=c/a;
          if (d<=0.0)  /*两个实根*/
          {   printf("X1=%f\n", sqrt(-d));
              printf("X2=%f\n", -sqrt(-d));
          }
          else  /*两个虚根*/
          {  printf("X1=+j%f\n", sqrt(d));
             printf("X2=-j%f\n", sqrt(d));
          }
   }
   else if (c==0.0)  /*方程为Ax²+Bx =0*/
   {      printf("X1=0.0\n");
          printf("X2=%f\n", -b/a);
   }
```

```
else  /*方程为Ax²+Bx+C =0*/
{   d=b*b-4*a*c;
    if (d>=0.0)
    /* B²−4AC≥0, 两个实根*/
    {  d=sqrt(d);
       if (b>0.0) x1=(-b-d)/(2*a);
       else  x1=(-b+d)/(2*a);
       x2=c/(a*x1);
       printf("X1=%f\n", x1);
       printf("X2=%f\n", x2);
    }
```

```
else /* B²−4AC <0, 两个共轭复根*/
{   d=sqrt(-d)/(2*a);
    p=-b/(2*a);
    printf("X1=%f+j%f\n", p, d);
    printf("X2=%f-j%f\n", p, d);
 }
}
}
```

运行结果:

```
input a, b, c: 1 1 1
X1=-0.500000+j0.866025
X2=-0.500000-j0.866025
请按任意键继续. . .
```

第5次作业

p.112~115

习题　2，4，7，11，14，19