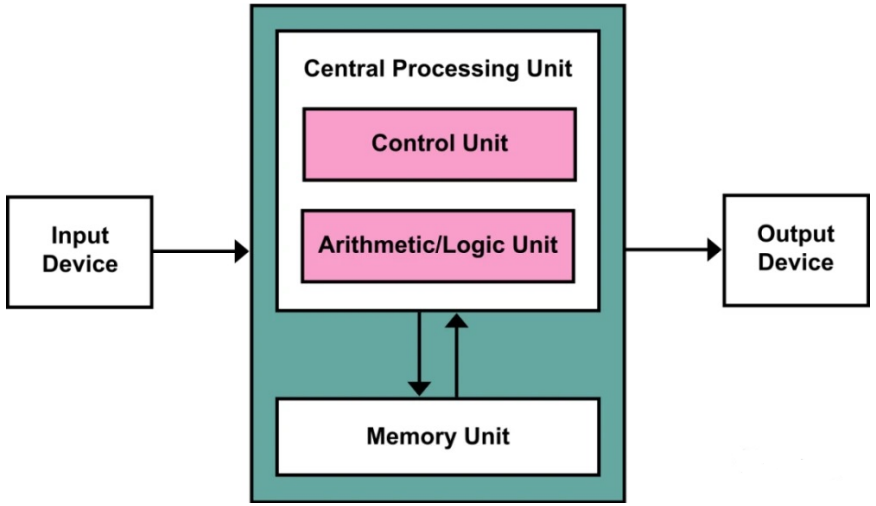
07 C控制语句 - 分支和跳转

内容提要

- if语句 和 if else语句
- 获得逻辑性
- 条件运算符？：
- 循环辅助手段：continue和break
- goto语句
- 关键概念



if语句

1 if语句

```
If(expression)
   statement
```

- if语句被称为分支语句（branching statement）或选择语句（selection statement）
  - 相当于一个交叉点，程序要在两条分支中选择一条执行
  - 如果expression值为真（非0），执行statement；否则就跳过该语句
- 和while语句的区别
  - if语句中，判定和执行仅有一次
  - while循环中，判断和执行可以多次

if语句

- 程序清单7.1　colddays.c
- 读取一列数据，每个数据都表示每日的最低温度（℃），然后打印统计的总天数和最低温度在0℃以下的天数占总天数的百分比
- while循环的测试条件利用scanf()的返回值来结束循环
  - scanf()在读到非数字字符时，会返回0
  - temperature的类型是float而不是int
- 为避免整数除法，将计算后的百分比强制转换为float类型

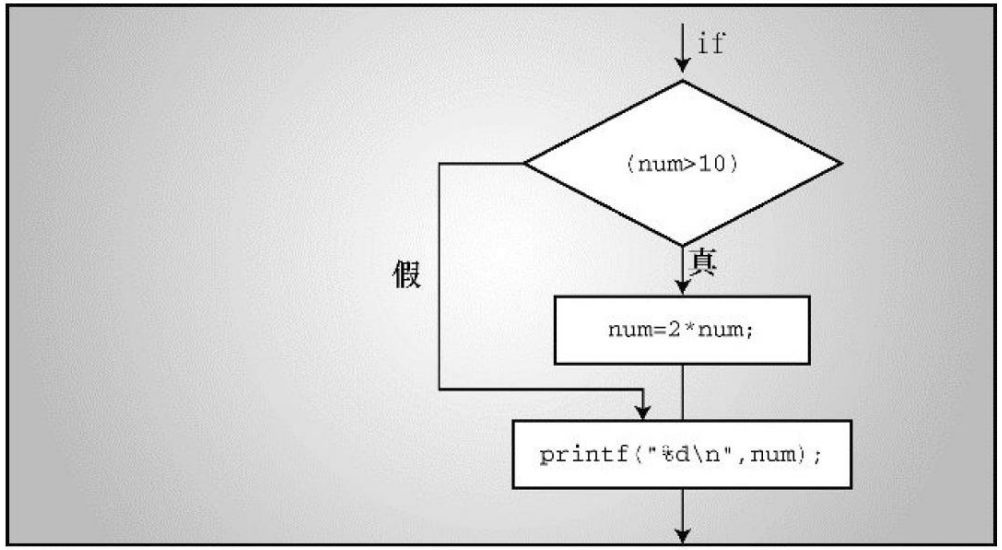
```
1.  #include <stdio.h>
2.  int main(void){
3.      const int FREEZING = 0;
4.      float temperature;
5.      int cold_days = 0;
6.      int all_days = 0;
7.
8.      printf("Enter the list of daily low temperatures.\n Use
    Celsius, and Q to quit.\n");
9.      while (scanf("%f", &temperature) == 1) {
10.         all_days++;
11.         if (temperature < FREEZING) cold_days++;
12.     }
13.     if (all_days != 0)
14.         printf("%d days total: %.1f%% were below freezing.\n",
    all_days, 100.0 * (float) cold_days / all_days);
15.     if (all_days == 0)
16.         printf("No data entered!\n");
17.     return 0;
18. }
```

if else语句

2 if else语句

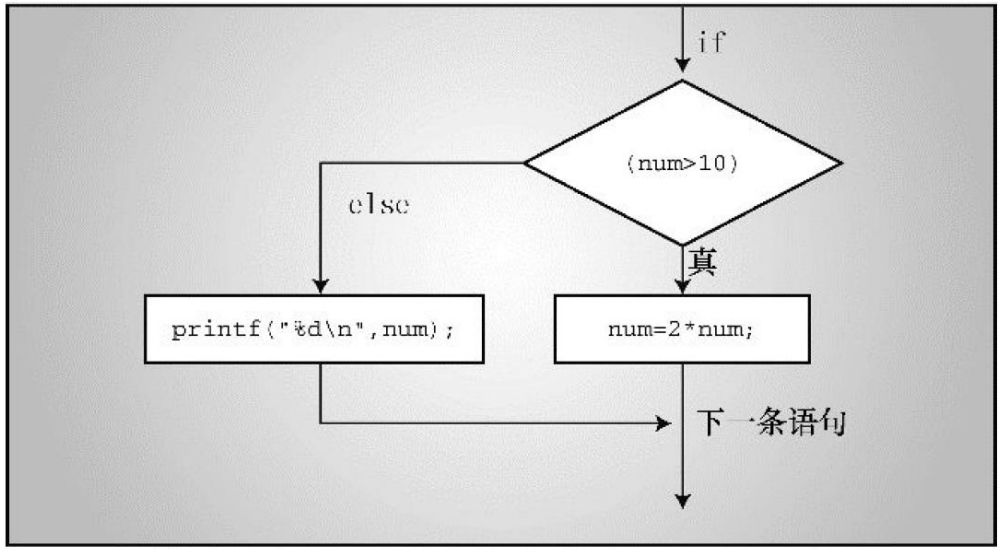
```
If(expression)
        statement1
else
        statement2
```

- expression为真(非0)，执行statement1
- expression为假或0，执行statement2



- if和else之间只允许一条语句（简单语句或复合语句）
  - 如if和else之间执行多条语句，需花括号

```
if (x > 0)
     printf("Incrementing x:\n");
     x++;
else          // 将产生一个错误
     printf("x <= 0 \n");
```



2.1 getchar()和putchar()

- getchar( )和putchar( )是面向字符的函数

- Getchar()没有参数，返回输入设备的下一个字符
  - ch = getchar(); //读取下一个字符并将值赋给ch
  - scanf(''%c‘’, ch); //效果一样

- putchar( ) 打印它的参数
  - putchar(ch); //先前赋给ch的值打印出来
  - printf(''%c‘’, ch); //效果一样

- 两个函数不需要格式说明符，只对字符起作用，比scanf()和printf() 更快更简洁

2.1 getchar()和putchar()

- 程序清单7.2　cypher1.c
- 把一行输入重新打印出来
  - 非空格替换成原字符在ASCII序列的下一个字符
  - 空格不变

```
1.  // cypher1.c -- alters input, preserving spaces
2.  #include <stdio.h>
3.  #define SPACE ' '   // that's quote-space-quote
4.  int main(void){
5.      char ch;
6.      ch = getchar();     // read a character
7.      while (ch != '\n')  // while not end of line
8.      {
9.          if (ch == SPACE)  // leave the space
10.             putchar(ch);  // character unchanged
11.         else
12.             putchar(ch + 1);  // other characters
13.         ch = getchar();       // get next character
14.     }
15.     putchar(ch);              // print the newline
16.
17.     return 0;
18. }
```

2.2 ctype.h系列字符函数

- ANSI C 有一系列标准的函数可以用来分析字符
  - ctype.h头文件包含了这些函数的原型
  - 函数接受一个字符作为参数，如果该字符属于某特定的种类则返回非零值（真）。否则返回零值（假）
- 程序清单7.3 cypher2.c

```
1.  // cypher2.c -- alters input, preserving non-letters
2.  #include <stdio.h>
3.  #include <ctype.h>              // for isalpha()
4.  int main(void)
5.  {
6.      char ch;
7.
8.      while ((ch = getchar()) != '\n')
9.      {
10.         if (isalpha(ch))       // if a letter,
11.             putchar(ch + 1);   // display next letter
12.         else                   // otherwise,
13.             putchar(ch);       // display as is
14.     }
15.     putchar(ch);               // display the newline
16.
17.     return 0;
18. }
```

ctype.h的字符判断函数

- 表7.1：字符测试函数
- 表7.2：字符映射函数

ctype.h的字符映射函数

- tolower( )
  - 如果参数是大写字母，返回相应的小写字母，否则返回原始参数
- toupper( )
  - 如果参数是小写字母，返回相应的大写字母，否则返回原始参数

- 映射函数并不改变原始的参数，只改变返回后的值，
- tolower(ch); //对ch没有影响
- ch = tolower(ch); //把ch转化为小写

2.3 多重选择else if

- 程序清单7.4 electric.c

- 用符号常量表示不同的费率和费率分界点
  - 把常量统一放在一处
  - 电力公司更新数据方便

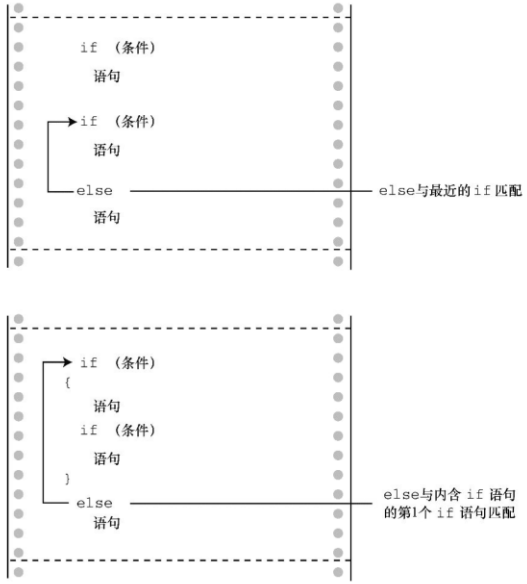
- else if是if else语句的变式
  - else { if(statement1) statement2}

- 花括号可以更清楚地表明这种特殊格式的含义

```
1.  #define RATE1   0.13230  // rate for first 360 kwh
2.  #define BREAK1  360.0    // first breakpoint
3.  #define BASE1   (RATE1 * BREAK1) // cost for 360
4.  int main(void){
5.      double kwh;  // kilowatt-hours used
6.      double bill;  // charges
7.      printf("Please enter the kwh used.\n");
8.      scanf("%lf", &kwh);         // %lf for type double
9.      if (kwh <= BREAK1)  bill = RATE1 * kwh;
10.     else if (kwh <= BREAK2)
11.         bill = BASE1 + (RATE2 * (kwh - BREAK1));
12.     else if (kwh <= BREAK3)
13.         bill = BASE2 + (RATE3 * (kwh - BREAK2));
14.     else
15.         bill = BASE3 + (RATE4 * (kwh - BREAK3));
16.     printf("The charge for %.1f kwh is $%1.2f.\n",
    kwh, bill);
17.     return 0;
18. }
```

2.4 else与if配对

- 如果程序中有许多if和else，如果没有花括号，else与离它最近的if匹配，除非最近的if被花括号括起来
- L1-L5
  - 实际上L4的else和L2的if匹配
  - 如果要实现缩进匹配的效果，需要修改成L6-12的形式




```
1.  if (number > 6)
2.      if (number < 12)
3.              printf("You're close!\n");
4.  else
5.      printf("Sorry, you lose a turn!\n");
```

```
6.  if (number > 6)
7.  {
8.      if (number < 12)
9.              printf("You're close!\n");
10. }
11. else
12.     printf("Sorry, you lose a turn!\n");
```

2.5 多层嵌套的if语句

- 程序清单7.5 divisors.c
- 给定一个整数，显示所有能整除它的约数。如果没有约数，则报告该数是一个素数（质数）

- 伪代码，设计如何找出约数
  - 循环检查2～num之间的所有数字
    - 测试它们是否能被num整除
    - 测试的数的范围：num的平方根
- isPrime
  - 标记（flag）变量
  - 判断
    - 只要有一个，就为false
    - 只有为true，才得到结论

```
1. int main(void){
2.     unsigned long num; // number to be checked
3.     unsigned long div; // potential divisors
4.     bool isPrime;      // prime flag
5.
6.     printf("Integer or q to quit.\n");
7.     while (scanf("%lu", &num) == 1){
8.         for (div = 2, isPrime = true; (div *
   div) <= num; div++){
9.             if (num % div == 0)
10.                isPrime = false; // not prime
11.        }
12.        if (isPrime)
13.           printf("%lu is prime.\n", num);
14.        printf("Another or q to quit.\n");
15.    }
16.    return 0;
17.}
```

逻辑运算符

3 逻辑运算符

- 把多个关系表达式组合起来

- 程序清单7.6　chcount.c
  - 计算输入的一行句子中除单引号和双引号以外其他字符的数量
- exp1和exp2是两个简单的关系表达式
  - 当且仅当exp1和exp2为真，exp1 && exp2为真
  - 如exp1或exp2为真，exp1 || exp2为真
  - 如exp1为假，!exp1为真；如exp1为真，!exp1为假
- 逻辑运算符的运算对象通常是关系表达式

```
1.  // chcount.c  -- use the logical AND operator
2.  #include <stdio.h>
3.  #define PERIOD '.'
4.  int main(void)
5.  {
6.      char ch;
7.      int charcount = 0;
8.
9.      while ((ch = getchar()) != PERIOD)
10.     {
11.         if (ch != '"' && ch != '\'')
12.             charcount++;
13.     }
14.     printf("There are %d ...\n", charcount);
15.
16.     return 0;
17. }
```

3.1 备选拼写：iso646.h头文件

- C99标准新增了可代替逻辑运算符的拼写
  - 定义在iso646.h头文件中
  - 如果在程序中包含该头文件，便可用and代替&&、or代替||、not代替!

3.2 优先级

- 优先级
  - !的优先级很高，高于乘法，和增量运算符优先级一样，仅次于圆括号
  - &&优先级高于||
  - 这两个运算符优先级都低于关系运算符而高于赋值运算

- 等价
  - `a > b && b > c || b > d`
  - `((a > b) && (b > c)) || (b > d)　　//推荐使用`
  - 推荐使用带圆括号的第2种写法。这样做即使不记得逻辑运算符的优先级，表达式的含义也很清楚

3.3 求值的顺序

- C把先计算哪部分的决定权留给编译器的设计者，以便针对特定系统优化设计
  - 但对于逻辑运算符是个例外，C保证逻辑表达式的求值顺序是从左往右。&&和||运算符都是序列点，所以程序在从一个运算对象执行到下一个运算对象之前，所有的副作用都会生效

- 例如：
  - `while（x++ <10 &&  x+y <20）`
  - 实际上，&&是一个序列点，这保证了在对&&右侧的表达式求值之前，已经递增了x

3.4 范围

- &&运算符用于测试范围
  - `if (range >= 90 && range <= 100)`
  - `if (90 <= range <= 100)  // 不可以!`
- 判断小写字符
  - `if (ch >= 'a' && ch <= 'z')`
  - `if (islower(ch))`

一个统计单词的程序

4 一个统计单词的程序

- 程序清单7.7 wordcnt.c
- 编写一个统计单词数量的程序
  - 读取并报告单词的数量，计算字符数和行数
  - 首先，逐个字符读取输入，知道何时停止读取
  - 然后，识别并计算：字符、行数和单词

读取一个字符

当有更多输入时

递增字符计数

如果读完一行，递增行数计数

如果读完一个单词，递增单词计数

读取下一个字符

```
1.  #define STOP '|'
2.  int main(void){
3.      char c;                 // read in character
4.      char prev;              // previous character read
5.      long n_chars = 0L;      // number of characters
6.      int n_lines = 0;        // number of lines
7.      int n_words = 0;        // number of words
8.      int p_lines = 0;        // number of partial lines
9.      bool inword = false;    // == true if c is in a word
10.
11.     printf("Enter text to be analyzed (| to terminate):\n");
12.     prev = '\n';            // used to identify complete lines
13.     while ((c = getchar()) != STOP) {
14.         n_chars++;          // count characters
15.         if (c == '\n') n_lines++;      // count lines
16.         if (!isspace(c) && !inword) {
17.             inword = true;  // starting a new word
18.             n_words++;      // count word
19.         }
20.         if (isspace(c) && inword) inword = false; // reached end of word
21.         prev = c;           // save character value
22.     }
23.     if (prev != '\n') p_lines = 1;
24.     printf("characters = %ld, words = %d, lines = %d, ", n_chars, n_words,
    n_lines);
25.     printf("partial lines = %d\n", p_lines);
26.
27.     return 0;
28. }
```

条件运算符？：

5 条件运算符？：

- 条件表达式（conditional expression）
  - 使用条件运算符，if else的简写方式
- expression1 ？expression2: expression3
  - expression1为真，表达式值为expression2
  - expression1为假，表达式值为expression3
- 条件运算符是C语言中唯一的三元运算符
- max = (a > b)? a:b;
  - 如果a 大于b,那么max等于a,否则等于b
- 程序清单7.8 paint.c

```c
/* paint.c -- uses conditional operator */
#include <stdio.h>
#define COVERAGE 350  // square feet per paint can
int main(void){
    int sq_feet;
    int cans;

    printf("Enter number of … be painted:\n");
    while (scanf("%d", &sq_feet) == 1)
    {
        cans = sq_feet / COVERAGE;
        cans += ((sq_feet % COVERAGE == 0)) ? 0 : 1;
        printf("You need %d %s of paint.\n", cans,
               cans == 1 ? "can" : "cans");
        printf("Enter next value (q to quit):\n");
    }

    return 0;
}
```

循环辅助手段：continue和break

6 循环辅助手段：continue和break

- continue语句
  - 可以用在三种循环方式
  - 运行到该句时，剩余迭代被忽略，开始下一次迭代
  - 如continue在嵌套结构中，仅影响最里层的结构
- 程序清单7.9 skippart.c
- 避免使用continue的方法【不必刻意】
- 执行continue语句后的下一个行为是继续“循环本体”语句
  - 对于while和do while，continue后的下一个行为是对循环的测试表达式求值
  - 对于for，continue后的下一个行为是对更新表达式求值，然后是对循环测试表达式求值

```
1.  int main(void){
2.      const float MIN = 0.0f;
3.      const float MAX = 100.0f;
4.      float score, total = 0.0f;
5.      int n = 0;
6.      float min = MAX, max = MIN;
7.      printf("Enter the first score (q to quit): ");
8.      while (scanf("%f", &score) == 1)    {
9.          if (score < MIN || score > MAX){
10.             printf("%0.1f is invalid. Try again: ", score);
11.             continue;// jumps to while loop test condition
12.         }
13.         printf("Accepting %0.1f:\n", score);
14.         min = (score < min)? score: min;
15.         max = (score > max)? score: max;
16.         total += score;
17.         n++;
18.         printf("Enter next score (q to quit): ");
19.     }
20.     return 0;
21. }
```

break语句

- 循环中的break语句
  - 退出当前循环，并进行程序的下一个阶段
- 在while和do while语句中，break语句使程序直接转到紧接着该循环后的第一条语句去执行
- 在for循环中，和continue不同，控制端的更新部分也将被跳过
- 嵌套里的break语句只是程序跳出里层的循环
- 7.10 break.c程序

```c
#include <stdio.h>
int main(void){
    float length, width;

    printf("Enter the length of the rectangle:\n");
    while (scanf("%f", &length) == 1) {
        printf("Length = %0.2f:\n", length);
        printf("Enter its width:\n");
        if (scanf("%f", &width) != 1)
            break;
        printf("Width = %0.2f:\n", width);
        printf("Area = %0.2f:\n", length * width);
        printf("Lof the rectangle:\n");
    }
    printf("Done.\n");

    return 0;
}
```

多重选择：switch和break

7 多重选择：switch和break

- switch的结构：

```
Switch(integer expression)
{
   case constant1:
       statements          //可选
   case constant:
       statements          //可选
   default :              //可选
       statements          //可选
}
```

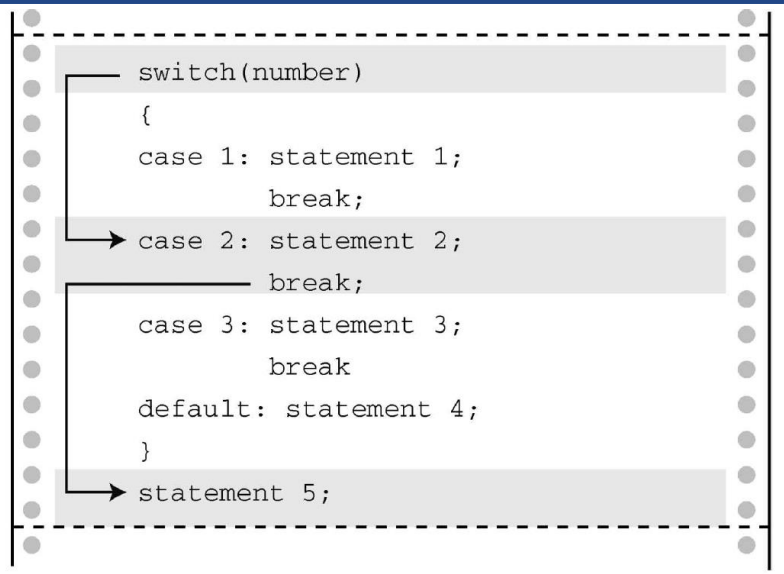
- 程序清单7.11 animals.c

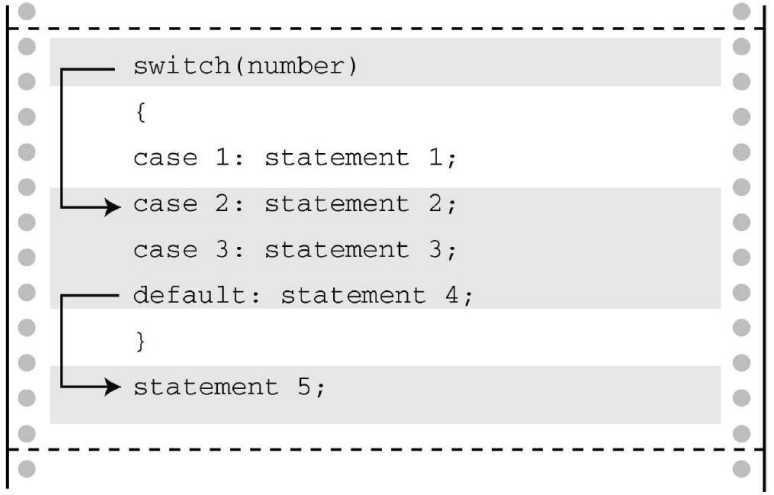
多重选择：switch和break

```c
int main(void){
    char ch;
    while ((ch = getchar()) != '#')    {
        if('\n' == ch) continue;
        if (islower(ch))   /* lowercase only */
            switch (ch){
            case 'a' :
                printf(“argali, … Asia\n");
                break;
            case 'b' :
                printf(“babirusa, … Malay\n");
                break;
            case 'c' :
                printf("coati, racoonlike mammal\n");
                break;
            case 'd' :
                printf("desman, aquatic, …
critter\n");
                break;
            case 'e' :
                printf("echidna, … anteater\n");
                break;
            case 'f' :
                printf("fisher, brownish marten\n");
                break;
            default :
                printf("That's a stumper!\n");
        } /* end of switch */
        else printf("I recognize … letters.\n");
        while (getchar() != '\n’) continue;
        printf(“Another letter or a #.\n");
    } /* while loop end  */
    return 0;
}
```

7.1 switch语句

- switch判断表达式具有整数值(包括char类型)
  - case标签是整型(及char)常量或者整数常量表达式
  - 不能用变量作为case标签
- switch后括号里的表达式被求值后，程序搜索一个与该值匹配的标签，然后跳到那行，若果没有匹配的，跳到被标记为default处
- 如没有break语句，从匹配标签开始执行到switch末尾
- 使用break会跳至结尾处





7.2 只读每行的首字符

- 程序清单7.11 animals.c
- 丢弃一行中其他字符的行为，经常出现在响应单字符的交互程序中
- 循环从输入中读取字符，包括按下Enter键产生的换行符
  - 注意，函数的返回值并没有赋给ch，以上代码所做的只是读取并丢弃字符
  - 由于最后丢弃的字符是换行符，所以下一个被读取的字符是下一行的首字母
  - 在外层while循环中，getchar()读取首字母并赋给ch

```
1. while (getchar() != '\n')
2.     continue; /* 跳过输入行的其余部分 */
```

7.3 多重标签

- 7.12 vowels.c程序
- 可以在switch语句中使用多重case标签
- 如果ch是字母i，switch语句会定位到标签为case 'i' :的位置。由于该标签没有关联break语句，所以程序流直接执行下一条语句，即i_ct++;。如果ch是字母I，程序流会直接定位到case 'I':
  - 本质上，两个标签都指的是相同的语句
- case 'U'的break语句并不需要。因为即使删除这条break语句，程序流会接着执行switch中的下一条语句，即default: break
  - 可把case ‘U’的break去掉以缩短代码【不推荐】

```
1.  int main(void){
2.      char ch;
3.      int a_ct, e_ct, i_ct, o_ct, u_ct;
4.      a_ct = e_ct = i_ct = o_ct = u_ct = 0;
5.      printf("Enter some text; enter # to quit.\n");
6.      while ((ch = getchar()) != '#'){
7.          switch (ch){
8.              case 'a' :
9.              case 'A' :  a_ct++;
10.                 break;
11.             case 'e' :
12.             case 'E' :  e_ct++;
13.                 break;
14.             default :   break;
15.         } // end of switch
16.     }     // while loop end
17.     return 0;
18. }
```

7.4 switch和if else

- 何时使用switch？何时使用if else？

- 如使用浮点类型的变量或表达式来选择，无法使用switch
  - 本质上要可数
- 如根据变量在某范围内决定程序流的去向，使用switch很麻烦，用if更方便
  - `if (integer < 1000 && integer > 2)`

goto语句

goto语句

- “谨慎使用，或者根本不用”

- goto语句有两部分：goto和标签名。标签的命名遵循变量命名规则

关键概念

关键概念

- 智能的一个方面是，根据情况做出相应的响应
  - 选择语句是开发具有智能行为程序的基础。C语言通过if、if else和switch语句，以及条件运算符（?:）实现智能选择
- if和if else语句使用测试条件来判断执行哪些语句
- 所有非零值都被视为true，零被视为false
  - 测试通常涉及关系表达式（比较两个值）、逻辑表达式（用逻辑运算符组合或更改其他表达式）
- 【通用原则】如要测试两个条件，使用逻辑运算符把两个完整的测试表达式组合起来
  - 正确：if（a<x && x<z）
  - 错误：if（a<x<z）

本章小结