# Divide and conquer.

C语言程序设计

C Programming

7

# C控制语句：分支和跳转

理论课程

厦門大學
XIAMEN UNIVERSITY

信息学院
(特色化示范性软件学院)
School of Informatics

黄 炜
博士，副教授
Wei Huang

## 内容要点

- 条件语句
- 跳转语句
- 多分支语句
- 常见错误

## 目录

# if 语句：格式

- 格式

```
if (<条件表达式>)
      <条件为真的语句（体）>
```

```
if (<条件表达式>)
      <条件为真的语句（体）>
else
      <条件为假的语句（体）>
```

  – if语句可以相互串联

```
if (<条件1表达式>)
      <条件1为真的语句（体）>
else if (<条件2表达式>)
      <条件2为真的语句（体）>
else
      <条件1和条件2为假的语句（体）>
```

```c
// colddays.c -- finds percentage of days below freezing
#include <stdio.h>
int main(void) {
    const int FREEZING = 0;
    float temperature;
    int cold_days = 0;
    int all_days = 0;
    printf("Enter the list (
    printf("Use Celsius, and enter q to quit.\n");
    while (scanf("%f", &temperature) == 1) {
        all_days++;
        if (temperature < FREEZING)
            cold_days++;
    }
    if (all_days != 0)
        printf("%d days total: %.1f%% were below freezing.\n",
               all_days, 100.0 * (float)cold_days / all_days);
    if (all_days == 0)
        printf("No data entered!\n");
    return 0;
}
```

```
Enter the list of daily low temperatures.
Use Celsius, and enter q to quit.
15↲
32↲
3↲
-6↲
-1↲
q↲
5 days total: 40.0% were below freezing.
```

此处强制类型转换没有必要

此处应该用else代替，更简洁

# if 语句：配对

- 配对：if-else语法
  - C语言的编译器没有缩进的概念
    - 空白字符是等效的
  - else与相邻上一个语句（或语句体）最近的上一个if配对

```
if (i<1)
     i++;
     i*=5;
else
     i*=3;
```

如果条件成立需要运行多条语句，应使用花括号

```
if (number > 6)
    if (number > 12)
        printf( "You're too close.\n" );
else
    printf( "Sorry, you lose a turn!\n" );
```

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

# 例题：字符替换加密

- 题目：替换型加密
  - 输入一行字符串，如果是空格则原样输出，其它情况ASCII码增加1输出。
  - 字母A换成B，B换成C，……
- 函数介绍

| 字符处理函数 | 格式化输入输出函数 |
|---|---|
| `ch=getchar();` | `scanf("%c", &ch);` |
| `putchar(ch);` | `printf("%c", ch);` |

```c
// cypher1.c -- alters input, preserving spaces
#include <stdio.h>
#define SPACE ' '                // that's quote-space-quote
int main(void) {
    char ch;
    ch = getchar();              // read a character
    while (ch != '\n')           // while not end of line
    {
        if (ch == SPACE)         // leave the space
            putchar(ch);         // character unchanged
        else
            putchar(ch + 1);     // change other characters
        ch = getchar();          // get next character
    }
    putchar(ch);                 // print the newline
    return 0;
}
```

相当于scanf("%c", &ch);

相当于printf("%c", ch);

```
This is a cat.↲
Uijt jt b dbu/
```

```c
// cypher2.c -- alters input, preserving non-letters
#include <stdio.h>
#include <ctype.h>              // for isalpha()
int main(void) {
    char ch;
    while ((ch = getchar()) != '\n')
    {
        if (isalpha(ch))        // if a letter,
            putchar(ch + 1);    // display next letter
        else                    // otherwise,
            putchar(ch);        // display as is
    }
    putchar(ch);                // display the newline

    return 0;
}
```

有经验的程序员将少数表达式嵌套合并在一起节省篇幅，但不应过长或过于复杂。

isalpha(ch)应包含ctype.h头文件，相当于
ch<='z' && ch>='a' || ch<='Z' && ch>='A'

```
This is a cat. That is a dog.↲
Uijt jt b dbu. Uibu jt b eph.
```

# 字符处理的主要函数

扩展内容

- ctype.h头文件的主要函数
  - 不推荐在编程中使用

| 函数 | 字符测试条件 | 函数 | 字符测试条件 |
|---|---|---|---|
| isalnum | 字母数字 | islower | 小写 |
| isalpha | 字母 | isprint | 可打印 |
| isblank | 空白 (空格或制表符) | ispunct | 标点 |
| iscntrl | 控制字符 (0x00 – 0x1F 或 0x7F) | isspace | 空白 |
| isdigit | 十进制数字 | isupper | 大写 |
| isgraph | 打印字符，除了空白以外 | isxdigit | 十六进制数 |

# 例题：分级收费

- 电力公司实行分级收费

| 级别 $(L_i)$ | $0 \sim 360$ | $360 \sim 468$ | $468 \sim 720$ | $720 \sim \infty$ |
|---|---|---|---|---|
| 费率 $(R_i)$ | 0.13230 | 0.15040 | 0.30025 | 0.34025 |

- 解题重点
  - 分段函数的构造

```c
// electric.c -- calculates electric bill
#include <stdio.h>
#define RATE1   0.13230      // rate for first 360 kwh
#define RATE2   0.15040      // rate for next 108 kwh
#define RATE3   0.30025      // rate for next 252 kwh
#define RATE4   0.34025      // rate for over 720 kwh
#define BREAK1  360.0        // first breakpoint for rates
#define BREAK2  468.0        // second breakpoint for rates
#define BREAK3  720.0        // third breakpoint for rates
#define BASE1   (RATE1 * BREAK1)  // cost for 360 kwh
#define BASE2   (BASE1 + (RATE2 * (BREAK2 - BREAK1)))
// cost for 468 kwh
#define BASE3   (BASE1 + BASE2 + (RATE3 *(BREAK3 -
BREAK2)))
//cost for 720 kwh
```

有经验的程序员将程序中的"配置"集中在代码的开始位置，便于读者修改配置，生成他们想要的程序

此处使用1、2、3作为后缀，是一级、二级、三级的意思，不写具体档位，是避免将来档位修改需要大改程序

```c
int main(void)
{
    double kwh;                         // kilowatt-hours used
    double bill;                        // charges
    printf("Please enter the kWh used.\n");
    scanf("%lf", &kwh);                 // %lf for type double
    if (kwh <= BREAK1)
        bill = RATE1 * kwh;
    else if (kwh <= BREAK2)             // kwh between 360 and 468
        bill = BASE1 + (RATE2 * (kwh - BREAK1));
    else if (kwh <= BREAK3)             // kwh betweent 468 and 720
        bill = BASE2 + (RATE3 * (kwh - BREAK2));
    else                                // kwh above 680
        bill = BASE3 + (RATE4 * (kwh - BREAK3));
    printf("The charge for %.1f kWh is $%1.2f.\n", kwh, bill);
    return 0;
}
```

当使用少量的实数时，默认使用double型，在输入时配合%lf

由于档位不多，没必要使用数组和循环进一步去耦合

```
Please enter the kWh used.
852↲
The charge for 852.0 kWh is $232.08.
```

## 例题：显示一个数的约数

- 解题重点
  - $a$在1到$n$中遍历，如果$a$能整除$n$，则$a$为$n$的约数
- 循环范围
  - 进一步缩小循环的起止范围，可以加快程序
    - $a \in \left[1, n\right]$? $a \in \left[1, \frac{n}{2}\right]$? $a \in \left[1, \sqrt{n}\right]$?
  - $a$自增1？自增2？

```c
// divisors.c -- nested ifs display divisors of a number
#include <stdio.h>
#include <stdbool.h>
int main(void) {
    unsigned long num;              // number to be checked
    unsigned long div;              // potential divisors
    bool isPrime;                   // prime flag
    printf("Please enter an integer for analysis; ");
    printf("Enter q to quit.\n");
    while (scanf("%lu", &num) == 1) {
        for (div = 2, isPrime = true; (div * div) <= num; div++) {
            if (num % div == 0) {
                if ((div * div) != num)
                    printf("%lu is divisible by %lu and %lu.\n",
                           num, div, num / div);
                else
                    printf("%lu is divisible by %lu.\n",
                           num, div);
                isPrime = false; // number is not prime
```

其实这里使用unsigned的必要性不大，一般也不写long而写int

有经验的程序员不用无意义的名字来命名变量

判断整数为0不宜简写为!(num % div)

默认设置isPrime为true，当找到了任何一个约数时设置为false

```c
            }
        }
        if (isPrime)
            printf("%lu is prime.\n", num);
        printf("Please enter another integer for analysis; ");
        printf("Enter q to quit.\n");
    }
    printf("Bye.\n");
    ret
}
```

此处判断变量是否为假，写为“isPrime!=0”反而误导读者

```
Please enter an integer for analysis; Enter q to quit.
13↲
13 is prime.
Please enter another integer for analysis; Enter q to quit.
4↲
4 is divisible by 2.
Please enter another integer for analysis; Enter q to quit.
2520↲
2520 is divisible by 2 and 1260.
2520 is divisible by 3 and 840.
2520 is divisible by 4 and 630.
（此处省略若干行）
Please enter another integer for analysis; Enter q to quit.
.↲
Bye.
```

# 例题：一个字符统计程序

- 读字符
- 当有字符输入时
  - 增加字符计数
  - 如果一行已读，增加行数（判断：回车符）
  - 如果一个单词已读，增加单词数（判断：空格）
  - 读字符
  - 返回循环

```c
// wordcnt.c -- counts characters, words, lines
#include <stdio.h>
#include <ctype.h>           // for isspace()
#include <stdbool.h>         // for bool, true, false
#define STOP '|'
int main(void)
{
    char c;                  // read in character
    char prev;               // previous character read
    long n_chars = 0L;       // number of characters
    int n_lines = 0;         // number of lines
    int n_words = 0;         // number of words
    int p_lines = 0;         // number of partial lines
    bool inword = false;     // == true if c is in a word

    printf("Enter text to be analyzed (| to terminate):\n");
    prev = '\n';         // used to identify complete lines
    while ((c = getchar()) != STOP) {
        n_chars++;           // count characters
```

使用is_inword较好

程序运行到此处暂停，直到用户输入回车继续运行，每次读一个字符，视情况进入循环，当所有字符全部读完，则继续等待输入。

```
        if (c == '\n')
            n_lines++;          // count lines
        if (!isspace(c) && !inword)  {
            inword = true;   // starting a new word
            n_words++;          // count word
        }
        if (isspace(c) && inword)
            inword = false; // reached end of word
        prev = c;               // save character value
    }
    if (prev != '\n')
        p_lines = 1;
    printf("characters = %ld, words = %d, lines = %d, ",
           n_chars, n_words, n_lines);
    printf("partial lines = %d\n", p_lines);
    return 0;
}
```

分情况讨论

如果未遇到'\n'，而是以'|'结尾，此时行数为1。

```
Enter text to be analyzed (| to terminate):
This is a cat.|↲
characters = 14, words = 4, lines = 0, partial lines = 1
```

# 目录

# 多分支：switch-case语句

- 语法
  - switch后只允许整型表达式
  - case后只允许确定值的整型表达式
    - 如：a-a，1……等
- 功能
  - 根据表达式值转到相应case标签
    - 类似goto
  - 除非遇到break语句，一直运行到switch语句结束
  - 如果没有找到合适的值，则转至default语句

把case看成标签

```
switch (<表达式>)
{
case <值1>:
      <语句（体）1>;
case <值2>:
      <语句（体）2>;
case <值3>:
      <语句（体）3>;
default:
      <语句（体）0>;
}
```

# 多分支：switch-case语句

```
switch (<表达式>)
{
case <值1>:
      <语句（体）1>;
      break;
case <值2>:
      <语句（体）2>;
      break;
case <值3>:
      <语句（体）3>;
      break;
default:
      <语句（体）0>;
}
<分支后语句（体）>;
```

```
switch (<表达式>)
{
case <值1>:
      <语句（体）1>;
case <值2>:
      <语句（体）2>;
case <值3>:
      <语句（体）3>;
default:
      <语句（体）0>;
}
<分支后语句（体）>;
```

```c
/* animals.c -- uses a switch statement */
#include <stdio.h>
#include <ctype.h>
int main(void) {
    char ch;
    printf("Give me a letter of the alphabet, and I will give ");
    printf("an animal name\nbeginning with that letter.\n");
    printf("Please type in a letter; type # to end my act.\n");
    while ((ch = getchar()) != '#') {
        if ('\n' == ch)
            continue;
        if (islower(ch))     /* lowercase only          */
            switch (ch) {
            case 'a':
                printf("argali, a wild sheep of Asia\n");
                break;
            case 'b':
                printf("babirusa, a wild pig of Malay\n");
                break;
            case 'c':
                printf("coati, racoonlike mammal\n");
                break;
```

这里只能是整型常数（字符型也是整型）

```
            case 'd':
                printf("desman, aquatic, molelike critter\n");
                break;
            c

            c

            default:
                printf("That's a stumper!\n");
            }                /* end of switch           */
        else
            printf("I recognize only lowercase letters.\n");
        while (getchar() != '\n')
            continue;      /* skip rest of input line */
        printf("Please type another letter or a #.\n");
    }                        /* while loop end          */
    printf("Bye!\n");

    return 0;
}
```

```
Give me a letter of the alphabet, and I will give an animal name
beginning with that letter.
Please type in a letter; type # to end my act.
I↲
I recognize only lowercase letters.
Please type another letter or a #.
#↲
Bye!
```

```c
// vowels.c -- uses multiple labels
#include <stdio.h>

int main(void) {
    char ch;
    int a_ct, e_ct, i_ct, o_ct, u_ct;

    a_ct = e_ct = i_ct = o_ct = u_ct = 0;

    printf("Enter some text; enter # to quit.\n");
    while ((ch = getchar()) != '#') {
        switch (ch) {
        case 'a':
        case 'A':  a_ct++;
            break;
        case 'e':
        case 'E':  e_ct++;
            break;
```

多个连续的case放在一起，可以将多个值指向同一语句

```
            case 'i' :
            case 'I' :  i_ct++;
                break;
            case 'o' :
            case 'O' :  o_ct++;
                break;
            case 'u' :
            case 'U' :  u_ct++;
                break;
            default :   break;
        }                           // end of switch
    }                               // while loop end
    printf("number of vowels:   A    E    I    O    U\n");
    printf("                        %4d %4d %4d %4d %4d\n",
           a_ct, e_ct, i_ct, o_ct, u_ct);
    return 0;
}
```

在默认标签后没有语句或只有break语句时，这两行语句可不写，不影响运行结果。

```
Enter some text; enter # to quit.
Joke#↲
number of vowels:   A    E    I    O    U
                    0    1    0    1    0
```

## 目 录

## 分支语句的常见错误

- 分支逻辑混乱，交叉覆盖或遗漏造成答案错误
  - 交叉覆盖时未使用else语句造成进入多个分支
- 逻辑“与”、“或”使用不当或优先级用错
- 将数组用排比if语句代替，实现和维护更复杂
- 多个if或循环语句嵌套导致缩进超过3层

## 目 录

C语言程序设计

C Programming

7

# 谢谢观看

理论课程

厦門大學
XIAMEN UNIVERSITY

信息学院
(特色化示范性软件学院)
School of Informatics

黄 炜
博士，副教授
Wei Huang