

Divide and conquer.



C控制语句： 分支和跳转

理论课程



廈門大學
XIAMEN UNIVERSITY



信息学院 黄 焯
(国家示范性软件学院) 博士, 副教授
School of Informatics Dr. Wei Huang

知识框架

- 条件语句
- 逻辑运算符
- 条件表达式
- 跳转语句
- 多分支语句
- 常见错误

内容纲要

1	条件语句
2	逻辑运算符
3	条件表达式
4	跳转语句
5	多分支语句

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 )>
```

```
// 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 of daily low temperatures.\n");
    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;
}

```

此处应该用else
代替，更简洁

此处强制类型转换
没有必要

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.

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

- 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。
 - 字母则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
<code>ch=getchar();</code>	<code>scanf("%c", &ch);</code>
<code>putchar(ch);</code>	<code>printf("%c", ch);</code>

```

// 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)
            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/

```
// 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;
}
```

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

- 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
 - 第一个360kWh，收费为\$0.12589/kWh
 - 下一个320kWh，收费为\$0.17901/kWh
 - 超过680kWh，收费为\$0.20791/kWh
- 解题重点
 - 分段函数的构造

```
// electric.c -- calculates electric bill
```

```
#include <stdio.h>
```

```
#define RATE1 0.13230
```

```
#define RATE2 0.15040
```

```
#define RATE3 0.30025
```

```
#define RATE4 0.34025
```

```
#define BREAK1 360.0
```

```
#define BREAK2 468.0
```

```
#define BREAK3 720.0
```

```
#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

// rate for next 252 kwh

// rate for over 720 kwh

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

```
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 between 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

- $a \in [1, n]$? $a \in [1, \frac{n}{2}]$? $a \in [1, \sqrt{n}]$?

- a 自增 1 ? 自增 2 ?


```
// divisors.c -- nested ifs display divisors of a number
```

```
#include <stdio.h>
```

```
#include <stdbool.h>
```

```
int main(void)
```

```
{
```

```
    unsigned long num;
```

```
    unsigned long div;
```

```
    bool isPrime;
```

```
    // number to be checked
```

```
    // potential divisors
```

```
    // prime flag
```

其实这里使用unsigned的必要性不大，一般也不写long而写int

```
    printf("Please enter an integer for analysis; ");
```

```
    printf("Enter q to quit.\n");
```

```
    while (scanf("%lu", &num) == 1)
```

```
    {
```

有经验的程序员不用毫无意义（如：flag）的名字来命名变量

```
        for (div = 2, isPrime = true; (div * div) <= num;
```

```
        div++)
```

```
        {
```

此处判断整数是否为0，简写为“!(num % div)”将误导读者

```
            if (num % div == 0)
```

```
            {
```

在不需要对约数排序时，可以使用成对的形式简化操作

此处判断变量是否为假，写为“isPrime!=0”反而误导读者

```
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

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

return 0;
}
```

初始状态设置isPrime为true，当找到了任何一个约数时设置为false

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.

98↵

98 is divisible by 2 and 49.

98 is divisible by 7 and 14.

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.

内容纲要

1	条件语句
2	逻辑运算符
3	条件表达式
4	跳转语句
5	多分支语句

逻辑运算的优先级与顺序

- 优先级

- 非 (!) : 优先级最高的 (相当于 “负号”)
- 与 (&&) : 优先级次之 (相当于 “乘法”)
- 或 (||) : 优先级再次之 (相当于 “加法”)

- 计算顺序

- 从左往右, “与” 左侧为0, 或 “或” 左侧**非0**, 不管之后怎么算结果都不会改变时, 右侧表达式不会被运算。

```
// chcount.c  -- use the logical AND operator
```

```
#include <stdio.h>
```

```
#define PERIOD '.'
```

```
int main(void)
```

```
{
```

```
    char ch;
```

```
    int charcount = 0;
```

```
    while ((ch = getchar()) != PERIOD)
```

```
    {
```

```
        if (ch != '"' && ch != '\\')
```

```
            charcount++;
```

```
    }
```

```
    printf("There are %d non-quote characters.\n", charcount);
```

```
    return 0;
```

```
}
```

注意单个变量在搭配
不等号时配合“与”，
反之在搭配等号时配
合“或”

I'm here. Where?

There are 7 non-quote characters.

取值范围的判断

- 用与和或来表示范围（相当于集合的交集和并集）

- `range > 90 && range <= 100` 表示 $\text{range} \in (90, 100]$
- `range > 90 || range <= 80` 表示 $\text{range} \in (-\infty, 80] \cup (90, +\infty)$
- 错误的表达式

`90 < range <= 100`

- 根据优先级顺序先算 `90 < range`，结果为0或1
- 再计算 `0 <= 100` 或 `1 <= 100`，结果一定为1

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


```

// 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

```

一般使用is_inword较好

```

while ((c = getchar()) != STOP) {
    n_chars++;
    if (c == '\n')
        n_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("Enter text to be analyzed (| to terminate):");
return This is a cat.|
}

```

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。

如果未遇到'\n'，而是以'|'结尾，此时行数为1。

characters = 14, words = 4, lines = 0, partial lines = 1

内容纲要

1	条件语句
2	逻辑运算符
3	条件表达式
4	跳转语句
5	多分支语句

三元运算符：条件运算符

- 条件表达式格式 `<表达式1>?<表达式2>:<表达式3>`
- 表达式的值
 - 当表达式1为0，其值为表达式3的值；否则为表达式2的值
 - 当表达式1为0，表达式2不被计算；否则表达式3不被计算
 - 条件表达式和表达式配合，if-else只能和语句配合

```
int max(int a, int b)
{
    return (a > b) ? a : b;
}
```

例题：一个喷漆程序

- 已知每罐油漆可喷的面积为350平方英尺
- 对任意输入的面积数目，推出需要的油漆罐数。

```
/* paint.c -- uses conditional operator */
```

```
#include <stdio.h>
```

```
#define COVERAGE 350
```

```
int main(void)
```

```
{
```

```
    int sq_feet;
```

```
    int cans;
```

```
    printf("Enter number of square feet to 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;
```

```
}
```

Enter number of square feet to be painted:

350

You need 1 can of paint.

Enter next value (q to quit):

700.3

You need 2 cans of paint.

Enter next value (q to quit):

这一部分简写为 `(sq_feet % COVERAGE != 0)`，是可以的

条件表达式中的表达式部分，
可以为字符串或其它表达式。

内容纲要

	2	逻辑运算符
	3	条件表达式
	4	跳转语句
	5	多分支语句
	6	常见编程错误

跳转：goto

- 一次返回多个循环可使用goto语句

– 滥用goto会使程序难以理解

```
while (exp1)
{
    while (exp2)
    {
        if (exp3)
        {
            goto out;
        }
    }
}
out:
```

```
ret = 0;
while (exp1)
{
    while (exp2)
    {
        if (exp3)
        {
            ret = 1;
            break;
        }
    }
    if (ret == 1)
        break;
}
```



```

/* skippart.c  -- uses continue to skip part of loop */
#include <stdio.h>
int main(void)
{
    const float MIN = 0.0f;
    const float MAX = 100.0f;
    float score;
    float total = 0.0f;
    int n = 0;
    float min = MAX;
    float max = MIN;

    printf("Enter the first score (q to quit): ");
    while (scanf("%f", &score) == 1)
    {
        if (score < MIN || score > MAX)
        {
            printf("%0.1f is an invalid value. Try again: ", score);
            continue; // jumps to while loop test condition
        }
    }
}

```

越界判断

```

}
printf("Accepting %0.1f:\n", score);
min = (score < min)? score: min;
max = (score > max)? score: max;
total += score;
n++;
printf("Enter next score (q to quit): ");

```

求最大（小）值的方法是将当前值与最大（小）值判断，如果更大（小），则记录下来

```

}
if (n > 0)
{

```

有经验的程序员在使用除数之前会判断除数是否为0

```

    printf("Average of %d scores is %0.1f.\n", n, total / n);
    printf("Low = %0.1f, high = %0.1f\n", min, max);

```

```

}
else

```

```

    printf("No valid scores were entered.\n");

```

```

return 0;

```

```

}

```

在程序无法得到正确答案时，程序应提供错误信息

```
Enter the first score (q to quit): 123↓  
123.0 is an invalid value. Try again: 59↓  
Accepting 59.0:  
Enter next score (q to quit): 100↓  
Accepting 100.0:  
Enter next score (q to quit): 85↓  
Accepting 85.0:  
Enter next score (q to quit): 83↓  
Accepting 83.0:  
Enter next score (q to quit): 0↓  
Accepting 0.0:  
Enter next score (q to quit): q↓  
Average of 5 scores is 65.4.  
Low = 0.0, high = 100.0
```

```

/* break.c -- uses break to exit a loop */
#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("Enter the length of the rectangle:\n");
    }
}

```

有经验的程序员在用户
输入错误时停止运算

```
printf("Done.\n");
```

```
return 0;
```

```
}
```

Enter the length of the rectangle:

15↓

Length = 15.00:

Enter its width:

30↓

Width = 30.00:

Area = 450.00:

Enter the length of the rectangle:

-3↓

Length = -3.00:

Enter its width:

9↓

Width = 9.00:

Area = -27.00:

Enter the length of the rectangle:

q↓

Done.

内容纲要

	3	条件表达式
	4	跳转语句
	5	多分支语句
	6	常见编程错误
	7	总结

多分支：switch-case语句

- 语法

把case看成是标签

- switch后只允许**整型表达式**
- case后只允许**整型常量**

- 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

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

```
switch (1 表达式>)  
{  
  case <值1>:  
    <语句 (体) 1>;  
  case <值2>:  
    <语句 2 体) 2>;  
  case <值3>:  
    <语句 3 本) 3>;  
  default:  
    <语句 4 本) 0>;  
}  
5 分支) 句 (体) >;
```



```

/* 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          */

```

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。

```
switch (ch)
{
```

这里只能是整型常数
(字符型也是整型)

```
    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;
    case 'd' :
        printf("desman, aquatic, molelike critter\n"); break;
    case 'e' :
        printf("echidna, the spiny 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 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");

```

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 #.

a

argali, a wild sheep of Asia

Please type another letter or a #.

f

fisher, brownish marten

Please type another letter or a #.

#

Bye!

```
// 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;

```

多个连续的case放在一起，
可以将多个值指向同一语句

```

case 'e' :
case 'E' :  e_ct++;
            break;
case 'i' :
case 'I' :  i_ct++;
            break;
case 'o' :
case 'O' :  o_ct++;
            break;
case 'u' :
case 'U' :  u_ct++;
            break;
default : break;

```

在默认标签之后没有语句
或只有break语句时，这些
语句可以不写，不影响运
行结果。

```

}
```

```

// 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;  
}
```

Enter some text; enter # to quit.

Joke#

number of vowels:	A	E	I	O	U
	0	1	0	1	0

内容纲要

	3	条件表达式
	4	跳转语句
	5	多分支语句
	6	常见编程错误
	7	总结

分支语句的常见错误

- 分支设计错误，没有正确覆盖造成答案错误
 - 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
- 排比使用的if语句没有换成switch或循环语句
- 多个if或循环语句嵌套导致缩进超过3层

内容纲要

	3	条件表达式
	4	跳转语句
	5	多分支语句
	6	常见编程错误
	7	总结

7

谢谢

理论课程



廈門大學
XIAMEN UNIVERSITY



信息学院 黄 焯
(国家示范性软件学院) 博士, 副教授
School of Informatics Dr. Wei Huang