C 语言

第八讲、字符输入输出与输入确认

张晓平

武汉大学数学与统计学院

2017 年 4 月 12 日

1. 一个统计字数的程序

2. getchar 与 putchar 函数

3. 缓冲区 (Buffer)

4. 终止键盘输入

5. 创建一个更友好的用户界面

6. 输入确认

7. 菜单浏览

1. 一个统计字数的程序

一个统计字数的程序

编制程序，读取一段文字，并报告其中的单词个数，同时统计字符个数和行数。

一个统计字数的程序

- 该程序应该逐个读取字符，并想办法判断何时停止。
- 应该能够识别并统计字符、行和单词。

一个统计字数的程序

```
1 // pseudo code
2  read a character
3  while there is more input
4        increment character count
5        if a line has been read, increment line
         count
6        if a word has been read, increment word
         count
7        read next character
```

一个统计字数的程序

```
// 循环输入结构
while ((ch = getchar()) != STOP)
{
   ...
}
```

一个统计字数的程序

```
// 循环输入结构
while ((ch = getchar()) != STOP)
{
   ...
}
```

在通用的单词统计程序中，换行符和句号都不适合标记一段文字的结束。我们将采用一个不常见的字符 |。

一个统计字数的程序

- 程序使用 getchar 来循环输入字符，可在每次循环通过递增一个字符计数器的值来统计字符。
- 为统计行数，程序可检查换行符。若字符为换行符，程序就递增行数计数器的值。若 STOP 字符出现在一行的中间，则将该行作为一个不完整行来统计，即该行有字符但没有换行符。

一个统计字数的程序

如何识别单词?

一个统计字数的程序

如何识别单词?

- 可将一个单词定义为不包含空白字符的一系列字符。
- 一个单词以首次遇到非空白字符开始，在下一个空白字符出现时结束。

一个统计字数的程序

- 检测非空白字符的判断表达式为

```
c != ' ' && c != '\n' && c != '\t'
```

或

```
!isspace(c) // #include <ctype.h>
```

- 检测空白字符的判断表达式为

```
c == ' ' || c == '\n' || c == '\t'
```

或

```
isspace(c) // #include <ctype.h>
```

一个统计字数的程序

- 为了判断一个字符是否在某个单词中，可在读入一个单词的首字符时把一个标志 (命名为 inword) 设置为 1，同时在此处递增单词个数。
- 只要 inword 为 1，后续的非空白字符就不标记为一个单词的开始。到出现下一个空白字符时，就把 inword 设置为 0。

```
1 // pseudo code
2    if c is not a whitespace and inword is false
3       set inword to true  and count the word
4    if c is a white space and inword is true
5       set inword to false
```

一个统计字数的程序 I

```
 1 // wordcnt.c:
 2 #include <stdio.h>
 3 #include <ctype.h>
 4 #include <stdbool.h>
 5 #define STOP '|'
 6 int main(void)
 7 {
 8   char c;
 9   char prev;
10   long n_chars = 0L;
11   int n_lines = 0;
12   int n_words = 0;
13   int p_lines = 0;
14   bool inword = false;
15   printf("Enter text (| to quit):\n");
```

一个统计字数的程序 II

```
16    prev = '\n';
17    while ((c = getchar()) != STOP) {
18      n_chars++;
19      if (c == '\n')
20        n_lines++;
21      if (!isspace(c) && !inword) {
22        inword = true;
23        n_words++;
24      }
25      if (isspace(c) && inword)
26        inword = false;
27      prev = c;
28    }
29    if (prev != '\n')
30      p_lines = 1;
```

一个统计字数的程序 III

```
31    printf("characters = %ld, words = %d, lines =
      %d, ", n_chars, n_words, n_lines);
32    printf("partial lines = %d\n", p_lines);
33    return 0;
34  }
```

一个统计字数的程序

```
Enter text (| to quit):
Reason is a
powerful servant but
an inadequate master.
|
characters = 56, words = 9, lines = 3, partial
lines = 0
```

2. getchar 与 putchar 函数

getchar 与 putchar 函数

```
1 // echo.c:
2 #include <stdio.h>
3 int main(void)
4 {
5   char ch;
6   while ((ch = getchar()) != '#
    ')
7     putchar(ch);
8   return 0;
9 }
```

getchar 与 putchar 函数

```
1 // echo.c:
2 #include <stdio.h>
3 int main(void)
4 {
5   char ch;
6   while ((ch = getchar()) != '#
    ')
7     putchar(ch);
8   return 0;
9 }
```

```
Hello world
Hello world
I am happy
I am happy
```

3. 缓冲区 (Buffer)

缓冲区 (Buffer)

- 非缓冲输入

  立即回显：键入的字符对正在等待的程序立即变为可用

  ```
  HHeellllloo wwoorrlldd[enter]
  II  aamm  hhaappppyy[enter]
  ```

- 缓冲输入

  延迟回显：键入的字符被存储在缓冲区中，按下回车键使字符块对程序变为可用。

缓冲区 (Buffer)：为什么需要缓冲区？

- 将若干个字符作为一个块传输比逐个发送耗时要少。
- 若输入有误，可以使用键盘来修正错误。当最终按下回车键时，便可发送正确的输入。

4. 终止键盘输入

终止键盘输入

程序 `echo.c` 在输入 `#` 时停止，但有一个问题，`#` 可能就是你想输入的字符。于是，我们自然希望终止字符不出现在文本中。

终止键盘输入：EOF

- C 让 `getchar` 在到达文件结尾时返回一个特殊值，其名称为 `EOF`(End Of File，文件结尾)。
- `scanf()` 在检测到文件结尾时也返回 `EOF`。
- `EOF` 在头文件 `stdio.h` 中定义

```
#define EOF (-1)
```

终止键盘输入：EOF 为什么是-1?

一般情况下，`getchar()` 返回一个 `0-127` 之间的值（标准字符集），或一个 `0-255` 的值（扩展字符集）。在两种情况下，`-1` 都不对应任何字符，故它可以表示文件结尾。

终止键盘输入：如何使用 EOF?

```
1 // echo_eof.c
2 #include <stdio.h>
3 int main(void)
4 {
5   int ch;
6   while ((ch = getchar()) != EOF)
7     putchar(ch);
8   return 0;
9 }
```

终止键盘输入：如何使用 EOF?

```
Hello world[enter]
Hello world[Ctrl+D]
```

终止键盘输入：如何使用 EOF?

要对键盘使用该程序，需要一种键入 EOF 的方式。

- 在大多数 Unix 系统上，在一行的开始位置键入 Ctrl+D 会导致传送文件尾信号。
- 其它系统中，可能将一行的开始位置键入的 Ctrl+Z 识别为文件尾信号，也可能把任意位置键入的 Ctrl+Z 识别为文件尾信号。

终止键盘输入：如何使用 EOF?

```
// Linux or Mac OS
Hello world[enter]
Hello world
[Ctrl+D]
```

5. 创建一个更友好的用户界面

创建一个更友好的用户界面

编制一个猜字程序，看是否为 1-100 之间的某个整数。程序会依次问你是否为 1、2、3、…，你回答 y 表示 yes，回答 n 表示 no，直到回答正确为止。

创建一个更友好的用户界面 I

```
1  // guess.c -- an inefficient and faulty number-
   guesser
2  #include <stdio.h>
3  int main(void)
4  {
5    int guess = 1;
6    printf("Pick an integer from 1 to 100. I will
     ");
7    printf("try to guess it.\nRespond with ");
8    printf("a y if my guess is right and with");
9    printf("\nan n if it is wrong.\n");
10   printf("Uh...is your number %d?\n", guess);
11   while (getchar() != 'y')
12     printf("Well, then, is it %d?\n", ++guess);
13   printf("I knew I could do it!\n");
```

创建一个更友好的用户界面 II

```
14    return 0;
15 }
```

创建一个更友好的用户界面

```
Pick an integer from 1 to 100. I will try to
guess it.
Respond with a y if my guess is right and with
an n if it is wrong.
Uh...is your number 1?
n
Well, then, is it 2?
Well, then, is it 3?
n
Well, then, is it 4?
Well, then, is it 5?
y
I knew I could do it!
```

创建一个更友好的用户界面

输入 n 时，竟然做了两次猜测，Why?

创建一个更友好的用户界面

输入 n 时，竟然做了两次猜测，Why?

是换行符，。。。，换行符在作怪。

创建一个更友好的用户界面

输入 n 时，竟然做了两次猜测，Why?

是换行符,。。。，换行符在作怪。

- 读入字符'n'，因'n' != 'y'，故打印

  ```
  Well, then, is it 2?
  ```

- 紧接着读入字符'\n'，因'\n' != 'y'，故打印

  ```
  Well, then, is it 3?
  ```

创建一个更友好的用户界面：解决方案

使用一个 while 循环来丢弃输入行的其它部分，包括换行符。

```
1 while (getchar() != 'y')
2 {
3   printf("Well, then, is it %d?\n", ++guess);
4   while (getchar() != '\n')
5     continue;  // skip rest of input line
6 }
```

这种处理办法还能把诸如 no 和 no way 这样的输入同简单的 n 一样看待。

创建一个更友好的用户界面

```
Pick an integer from 1 to 100. I will try to
guess it.
Respond with a y if my guess is right and with
an n if it is wrong.
Uh...is your number 1?
n
Well, then, is it 2?
no
Well, then, is it 3?
no sir
Well, then, is it 4?
forget it
Well, then, is it 5?
y
I knew I could do it!
```

创建一个更友好的用户界面

若不希望将 f 的含义看做与 n 相同，可使用一个 if 语句来筛选掉其它响应。

```
1  char ch;
2  ...
3  while ((ch = getchar()) != 'y')
4  {
5    if (ch == 'n')
6      printf("Well, then, is it %d?\n", ++guess);
7    else
8      printf("Sorry, I understand only y or n.\n")
       ;
9    while (getchar() != '\n')
10     continue;  // skip rest of input line
11 }
```

创建一个更友好的用户界面

```
Pick an integer from 1 to 100. I will try to
guess it.
Respond with a y if my guess is right and with
an n if it is wrong.
Uh...is your number 1?
n
Well, then, is it 2?
no
Well, then, is it 3?
no sir
Well, then, is it 4?
forget it
Sorry, I understand only y or n.
n
Well, then, is it 5?
y
I knew I could do it!
```

创建一个更友好的用户界面

当你编写交互式程序时，应试着去预料用户未能遵循指示的可能方式，然后让程序能合理地处理用户的疏忽。并告诉用户哪里出了错误，给予他们另一次机会。

创建一个更友好的用户界面：混合输入数字和字符

若你的程序需要使用 `getchar()` 输入字符和使用 `scanf()` 输入数字。两个函数都很很好的独立完成其工作，但不能很好的混合在一起。因为

- `getchar()` 读取每个字符，包括空格、制表符和换行符
- `scanf()` 在读取数字时会跳过空格、制表符和换行符。

创建一个更友好的用户界面：混合输入数字和字符

编写程序，读取一个字符和两个整数，然后使用这两个整数指定行数和列数打印该字符。

创建一个更友好的用户界面：混合输入数字和字符 I

```
1  /* showchar1.c -- program with a BIG I/O problem
     */
2  #include <stdio.h>
3  void display(char cr, int lines, int width);
4  int main(void)
5  {
6    int ch; /* character to be printed */
7    int rows, cols; /* number of rows and columns
     */
8    printf("Enter a character and two integers:\n"
     );
9    while ((ch = getchar()) != '\n') {
10     scanf("%d %d", &rows, &cols);
11     display(ch, rows, cols);
```

创建一个更友好的用户界面：混合输入数字和字符 II

```
12      printf("Enter another character and two
        integers;\n");
13      printf("Enter a newline to quit.\n");
14    }
15    printf("Bye.\n");
16    return 0;
17  }
18
19  void display(char cr, int lines, int width)
20  {
21    int row, col;
22    for (row = 1; row<= lines; row++)
23    {
24      for (col = 1; col <= width; col++)
25        putchar(cr);
26      putchar('\n');
```

创建一个更友好的用户界面：混合输入数字和字符 III

```
27      }
28  }
```

创建一个更友好的用户界面：混合输入数字和字符

```
Enter a character and two integers:
c 2 3[enter]
ccc
ccc
Enter another character and two integers;
Enter a newline to quit.
Bye.
```

创建一个更友好的用户界面：混合输入数字和字符

- 程序开始时表现很好。当你输入 c 2 3 时，如期打印 2 行 3 列 c 字符。
- 然后程序提示输入第二组数据，但还没等你输入程序就退出了。Why?

创建一个更友好的用户界面：混合输入数字和字符

- 程序开始时表现很好。当你输入 c 2 3 时，如期打印 2 行 3 列 c 字符。
- 然后程序提示输入第二组数据，但还没等你输入程序就退出了。Why?

  又是它在作怪!

创建一个更友好的用户界面：混合输入数字和字符

- 程序开始时表现很好。当你输入 c 2 3 时，如期打印 2 行 3 列 c 字符。
- 然后程序提示输入第二组数据，但还没等你输入程序就退出了。Why?

  又是它在作怪！谁?

创建一个更友好的用户界面：混合输入数字和字符

- 程序开始时表现很好。当你输入 c 2 3 时，如期打印 2 行 3 列 c 字符。
- 然后程序提示输入第二组数据，但还没等你输入程序就退出了。Why?

  又是它在作怪！谁？换行符！

创建一个更友好的用户界面：混合输入数字和字符

- 程序开始时表现很好。当你输入 c 2 3 时，如期打印 2 行 3 列 c 字符。
- 然后程序提示输入第二组数据，但还没等你输入程序就退出了。Why?

  又是它在作怪！谁？换行符！
- 输入第一组数据后按下了换行符，scanf 将它留在了输入队列。

  而 `getchar()` 并不跳过换行符，故在下一次循环时，`getchar()` 读取了该字符，并将其值赋给了 ch，而 ch 为换行符正是终止循环的条件。

创建一个更友好的用户界面：解决方案

- 程序必须跳过一个输入周期中最后一个数字与下一行开始出键入的字符之间的所有换行符或空格。
- 若除了 `getchar()` 判断之外还可以在 `scanf()` 阶段终止程序，则会更好。

创建一个更友好的用户界面：混合输入数字和字符 I

```
1  /* showchar2.c -- prints characters in rows and
   columns */
2  #include <stdio.h>
3  void display(char cr, int lines, int width);
4  int main(void)
5  {
6    int ch; /* character to be printed */
7    int rows, cols; /* number of rows and columns
     */
8    printf("Enter a character and two integers:\n"
     );
9    while ((ch = getchar()) != '\n') {
10     if (scanf("%d %d",&rows, &cols) != 2) break;
11     display(ch, rows, cols);
12     while (getchar() != '\n')
```

创建一个更友好的用户界面：混合输入数字和字符 II

```
13          continue;
14        printf("Enter another character and two
          integers;\n");
15        printf("Enter a newline to quit.\n");
16      }
17      printf("Bye.\n");
18      return 0;
19  }
20
21  void display(char cr, int lines, int width)
22  {
23      int row, col;
24      for (row = 1; row<= lines; row++)
25      {
26          for (col = 1; col <= width; col++)
27              putchar(cr);
```

创建一个更友好的用户界面：混合输入数字和字符 III

```
28        putchar('\n');
29    }
30 }
```

创建一个更友好的用户界面：混合输入数字和字符

```
Enter a character and two integers:
c 1 3[enter]
ccc
Enter another character and two integers;
Enter a newline to quit.
! 3 6[enter]
!!!!!!
!!!!!!
!!!!!!
Enter another character and two integers;
Enter a newline to quit.

Bye.
```

6. 输入确认

输入确认

- 在实际情况中，用户并不总是遵循指令，在程序所希望的输入与其实际输入之间可能存在不匹配，这可能会导致程序运行失败。
- 作为程序员，你应该预见所有可能的输入错误，修正程序以使其能检测到这些错误并作出处理。

输入确认

1、如有一个处理非负数的循环，用户可能会输入一个负数，你可以用一个关系表达式来检测这类错误：

```
int n;
scanf("%d", &n); // get first value
while (n >= 0)   // detect out-of-range value
{
  // process n
  scanf("%d", &n); // get next value
}
```

输入确认

2、当然用户还可能输入类型错误的值，如字符 q。检测这类错误的方式是检测 scanf 的返回值。

该函数返回成功读入的项目个数，因此仅当用户输入一个整数时，下列表达式为真：

```
scanf("%d", &n) == 1
```

输入确认

考虑以上两种可能出现的输入错误，我们可以对代码进行改进：

```
int n;
while (scanf("%d", &n) == 1 && n >= 0)
{
    // process n
}
```

while 循环的条件是“当输入是一个整数并且该整数为正”。

输入确认

上面的例子中，当输入类型有错时，则终止输入。而更合适的处理方式是让程序对用户更加友好，给用户尝试输入正确类型的机会。

- 首先要剔除那些有问题的输入，因 scanf 没有成功读取输入，会将其留在输入队列中。
- 然后使用 `getchar()` 来逐个字符地读取输入。

输入确认

编制程序，计算特定范围内所有整数的平方和。限制这个特定范围的上届不应大于 1000，下界不应小于-1000。

输入确认 I

```
1  /* checking.c -- validating input */
2  #include <stdio.h>
3  #include <stdbool.h>
4  // validate that input is an integer
5  int get_int(void);
6  // validate that range limits are valid
7  bool bad_limits(int begin, int end, int low, int
    high);
8  // calculate the sum of the squares of the
   integer a through b
9  double sum_squares(int a, int b);
10 int main(void)
11 {
12   const int MIN = -1000;
13   const int MAX = +1000;
```

输入确认 II

```
14    int start;
15    int stop;
16    double answer;
17    printf("This program computes the sum of the "
18           "squares of integers in a range.\n"
19           "The lower bound should not be less
      than "
20           "-1000 and\nthe upper bound should not
      "
21           "be more than +1000.\nEnter the limits
      "
22           "(enter 0 for both limits to quit):\n"
23           "lower limit: ");
24    start = get_int();
25    printf("upper limit: ");
26    stop = get_int();
```

输入确认 III

```
27    while (start !=0 || stop != 0) {
28      if (bad_limits(start, stop, MIN, MAX))
29        printf("Please try again.\n");
30      else {
31        answer = sum_squares(start, stop);
32        printf("The sum of the squares of the
          integers ");
33        printf("from %d to %d is %g\n", start,
          stop, answer);
34      }
35      printf("Enter the limits (enter 0 for both "
36             "limits to quit):\n");
37      printf("lower limit: ");
38      start = get_int();
39      printf("upper limit: ");
40      stop = get_int();
```

输入确认 IV

```
41    }
42    printf("Done.\n");
43    return 0;
44  }
45  int get_int(void)
46  {
47    int input;
48    char ch;
49    while (scanf("%d", &input) != 1) {
50      while ((ch = getchar()) != '\n')
51        putchar(ch); // dispose of bad input
52      printf(" is not an integer.\n");
53      printf("Please enter an integer value, ");
54      printf("such as 25, -178, or 3: ");
55    }
56    return input;
```

输入确认 V

```
57 }
58 double sum_squares(int a, int b) {
59   double total = 0;
60   int i;
61   for (i = a; i <= b; i++)
62     total += i * i;
63   return total;
64 }
65 bool bad_limits(int begin, int end, int low, int
    high)
66 {
67   bool not_good = false;
68   if (begin > end)
69   {
70     printf("%d isn't smaller than %d.\n",
71            begin, end);
```

输入确认 VI

```
72      not_good = true;
73    }
74    if (begin < low || end < low) {
75      printf("Values must be %d or greater.\n",
76             low);
77      not_good = true;
78    }
79    if (begin > high || end > high) {
80      printf("Values must be %d or less.\n",
81             high);
82      not_good = true;
83    }
84    return not_good;
85  }
```

输入确认

对于 `get_int()`，

- 该函数试图将一个 int 值读入变量 input。
- 若失败，则该函数进入外层 while 循环，然后内层 while 循环逐个字符地读取那些有问题的输入字符。
- 然后该函数提示用户重新尝试。外层循环继续运行，直至用户成功地输入一个整数。

输入确认

对于 `bad_limits()`，用户输入一个下界和上界来定义值域。需要的检查可能有

- 第一个值是否小于等于第二个值;
- 两个值是否在可接受的范围内。

输入确认 I

```
This program computes the sum of the squares of
integers in a range.
The lower bound should not be less than -1000
and
the upper bound should not be more than +1000.
Enter the limits (enter 0 for both limits to
quit):
lower limit: 1q
upper limit: q is not an integer.
Please enter an integer value, such as 25, -178,
 or 3: 3
The sum of the squares of the integers from 1 to
 3 is 14
Enter the limits (enter 0 for both limits to
quit):
```

输入确认 II

```
lower limit: q
q is not an integer.
Please enter an integer value, such as 25, -178,
 or 3: 3
upper limit: 5
The sum of the squares of the integers from 3 to
 5 is 50
Enter the limits (enter 0 for both limits to
quit):
lower limit: 4
upper limit: 3q
4 isn't smaller than 3.
Please try again.
Enter the limits (enter 0 for both limits to
quit):
lower limit: q is not an integer.
```

输入确认 III

```
Please enter an integer value, such as 25, -178,
 or 3: 0
upper limit: 0
Done.
```

输入确认：模块化编程

使用独立的函数来实现不同的功能。程序越大，模块化编程就越重要。

- main 函数管理流程，为其它函数指派任务；
- get_int 函数获取输入；
- badlimits 函数检查值的有效性；
- sum_squares 函数进行实际的计算。

输入确认：C 输入的工作方式

假如有输入

```
is 28 12.4
```

在你看来，该输入是一串字符、一个整数、一个浮点值。而对 C 来说，该输入时一个字节流。

- 第 1 个字节是字母 i 的字符编码
- 第 2 个字节是字母 s 的字符编码
- 第 3 个字节是空格字符的字符编码
- 第 4 个字节是数字 2 的字符编码
- ...

输入确认：C 输入的工作方式

当 `getchar()` 遇到这一行，以下代码将读取并丢弃整行，包括数字，因为这些数字其实被看做是字符：

```
while((ch = getchar()) != '\n')
  putchar(ch);
```

输入确认：C 输入的工作方式

假如有输入

```
42
```

在使用 `scanf()` 函数时，不同的占位符会导致不同的效果。

输入确认：C 输入的工作方式

- 使用`%c`，将只读取字符`'4'`并将其存储在一个 char 型变量中；
- 使用`%s`，会读取两个字符，即字符`'4'`和`'2'`，并将它们存储在一个字符串中
- 使用`%d`，同样读取两个字符，但随后会计算与它们相应的整数值 $4 \times 10 + 2 = 42$，然后将该整数保存在一个 int 变量中；
- 使用`%f`，同样读取两个字符，计算对应的数值 `42`，然后以浮点表示法表示该值，并将结果保存在一个 float 型变量中。

7. 菜单浏览

菜单浏览

菜单作为用户界面的一部分，会使程序对用户更友好，但也给程序员提出了一些新问题。

```
Enter the letter of your choice:
a. advice          b. bell
c. count           q. quit
```

编程目标:

- 让程序在用户遵循指令时顺利进行
- 让程序在用户没有遵循指令时也能顺利进行

菜单浏览

编写程序，确保有如下输出：

菜单浏览

```
Enter the letter of your choice:
a. advice     b. bell
c. count      q. quit
a[enter]
Buy low, sell high.
Enter the letter of your choice:
a. advice     b. bell
c. count      q. quit
b[enter]
Enter the letter of your choice:
a. advice     b. bell
c. count      q. quit
c[enter]
Count how far? Enter an integer:
two[enter]
two is not an integer.
```

菜单浏览

```
Please enter an integer value,
such as 25, -178, or 3: 5[enter]
1
2
3
4
5
Enter the letter of your choice:
a. advice     b. bell
c. count      q. quit
q
Bye.
```

菜单浏览 I

```
 1 /* menu.c -- menu techniques */
 2 #include <stdio.h>
 3 char get_choice(void);
 4 char get_first(void);
 5 int get_int(void);
 6 void count(void);
 7 int main(void)
 8 {
 9   int choice;
10   while ( (choice = get_choice()) != 'q') {
11     switch (choice) {
12     case 'a': printf("Buy low, sell high.\n");
13       break;
14     case 'b': putchar('\a');
15       break;
```

菜单浏览 II

```
16      case 'c': count();
17        break;
18      default : printf("Program error!\n");
19        break;
20      }
21    }
22    printf("Bye.\n");
23    return 0;
24  }
25
26  void count(void)
27  {
28    int n, i;
29    printf("Count how far? Enter an integer:\n");
30    n = get_int();
31    for (i = 1; i <= n; i++)
```

菜单浏览 III

```
32      printf("%d\n", i);
33    while ( getchar() != '\n')
34          continue;
35  }
36
37  char get_choice(void)
38  {
39    int ch;
40    printf("Enter the letter of your choice:\n");
41    printf("a. advice     b. bell\n");
42    printf("c. count      q. quit\n");
43    ch = get_first();
44    while( (ch<'a' || ch>'c') && ch!='q')
45    {
46      printf("Please respond with a, b, c, or q.\n
      ");
```

菜单浏览 IV

```
47      ch = get_first();
48    }
49    return ch;
50  }
51
52  char get_first(void)
53  {
54    int ch;
55    ch = getchar();
56    while (getchar() != '\n')
57      continue;
58    return ch;
59  }
60
61  int get_int(void)
62  {
```

菜单浏览 V

```
63    int input;
64    char ch;
65    while (scanf("%d", &input) != 1) {
66      while ((ch = getchar()) != '\n')
67        putchar(ch); // dispose of bad input
68      printf(" is not an integer.\n");
69      printf("Enter an integer value,\n");
70      printf("such as 25, -178, or 3: ");
71    }
72    return input;
73  }
```