C/C++

字符输入输出和字符确认

张晓平

武汉大学数学与统计学院

Table of contents

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

```c
// wordcnt.c:
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#define STOP '|'
int main(void)
{
  char c, prev;
  long n_chars = 0L;
  int n_lines = 0, n_words = 0, p_lines = 0;
  bool inword = false;
  printf("Enter text (| to quit):\n");
  prev = '\n';
  while ((c = getchar()) != STOP) {
    n_chars++;
    if (c == '\n') n_lines++;
```

一个统计字数的程序 II

```
    if (!isspace(c) && !inword) {
      inword = true;
      n_words++;
    }
    if (isspace(c) && inword) inword = false;
    prev = c;
  }
  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;
}
```

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
```

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
```

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。

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?

```c
// echo_eof.c
#include <stdio.h>
int main(void)
{
  int ch;
  while ((ch = getchar()) != EOF)
    putchar(ch);
  return 0;
}
```

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]
```

终止键盘输入

8.4 重定向与文件

8.4 重定向与文件

创建一个更友好的用户界面

创建一个更友好的用户界面

例

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

```c
// guess.c -- an inefficient and faulty number-guesser
#include <stdio.h>
int main(void)
{
  int guess = 1;
  printf("Pick an integer from 1 to 100. I will ");
  printf("try to guess it.\nRespond with ");
  printf("a y if my guess is right and with");
  printf("\nan n if it is wrong.\n");
  printf("Uh...is your number %d?\n", guess);
  while (getchar() != 'y')
    printf("Well, then, is it %d?\n", ++guess);
  printf("I knew I could do it!\n");
  return 0;
}
```

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?

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?

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。

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

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!
```

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

例

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界不应大于 1000，下界不应小于 –1000。

输入确认 I

```c
/* checking.c -- validating input */
#include <stdio.h>
#include <stdbool.h>
// validate that input is an integer
int get_int(void);
// validate that range limits are valid
bool bad_limits(int begin, int end, int low, int
 high);
// calculate the sum of the squares of the
integer a through b
double sum_squares(int a, int b);
int main(void)
{
  const int MIN = -1000;
```

输入确认 II

```
const int MAX = +1000;
int start;
int stop;
double answer;
printf("This program computes the sum of the "
       "squares of integers in a range.\n"
       "The lower bound should not be less
       than "
       "-1000 and\nthe upper bound should not
       "
       "be more than +1000.\nEnter the limits
       "
       "(enter 0 for both limits to quit):\n"
       "lower limit: ");
start = get_int();
```

输入确认 III

```
printf("upper limit: ");
stop = get_int();
while (start !=0 || stop != 0) {
  if (bad_limits(start, stop, MIN, MAX))
    printf("Please try again.\n");
  else {
    answer = sum_squares(start, stop);
    printf("The sum of the squares of the
    integers ");
    printf("from %d to %d is %g\n", start,
    stop, answer);
  }
  printf("Enter the limits (enter 0 for both "
         "limits to quit):\n");
  printf("lower limit: ");
```

输入确认 IV

```
    start = get_int();
    printf("upper limit: ");
    stop = get_int();
  }
  printf("Done.\n");
  return 0;
}
int get_int(void)
{
  int input;
  char ch;
  while (scanf("%d", &input) != 1) {
    while ((ch = getchar()) != '\n')
      putchar(ch); // dispose of bad input
    printf(" is not an integer.\n");
```

输入确认 V

```
    printf("Please enter an integer value, ");
    printf("such as 25, -178, or 3: ");
  }
  return input;
}
double sum_squares(int a, int b) {
  double total = 0;
  int i;
  for (i = a; i <= b; i++)
    total += i * i;
  return total;
}
bool bad_limits(int begin, int end, int low, int
 high)
{
```

输入确认 VI

```
bool not_good = false;
if (begin > end)
{
  printf("%d isn't smaller than %d.\n",
         begin, end);
  not_good = true;
}
if (begin < low || end < low) {
  printf("Values must be %d or greater.\n",
         low);
  not_good = true;
}
if (begin > high || end > high) {
  printf("Values must be %d or less.\n",
         high);
```

输入确认 VII

```
    not_good = true;
  }
  return not_good;
}
```

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):
lower limit: q
q is not an integer.
Please enter an integer value, such as 25, -178, or 3:
 3
```

输入确认 II

```
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.
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()` 管理流程，为其它函数指派任务；
- `get_int()` 获取输入；
- `badlimits()` 检查值的有效性；
- `sum_squares()` 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。

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

```c
/* menu.c -- menu techniques */
#include <stdio.h>
char get_choice(void);
char get_first(void);
int get_int(void);
void count(void);
int main(void)
{
  int choice;
  while ( (choice = get_choice()) != 'q') {
    switch (choice) {
    case 'a': printf("Buy low, sell high.\n"); break;
    case 'b': putchar('\a'); break;
    case 'c': count(); break;
    default : printf("Program error!\n"); break;
    }
```

菜单浏览 II

```
  }
  printf("Bye.\n");
  return 0;
}

void count(void)
{
  int n, i;
  printf("Count how far? Enter an integer:\n");
  n = get_int();
  for (i = 1; i <= n; i++)
    printf("%d\n", i);
  while ( getchar() != '\n') continue;
}

char get_choice(void)
```

菜单浏览 III

```
{
  int ch;
  printf("Enter the letter of your choice:\n");
  printf("a. advice     b. bell\n");
  printf("c. count      q. quit\n");
  ch = get_first();
  while( (ch<'a' || ch>'c') && ch!='q') {
    printf("Please respond with a, b, c, or q.\n");
    ch = get_first();
  }
  return ch;
}

char get_first(void)
{
  int ch;
```

菜单浏览 IV

```
  ch = getchar();
  while (getchar() != '\n') continue;
  return ch;
}

int get_int(void)
{
  int input;
  char ch;
  while (scanf("%d", &input) != 1) {
    while ((ch = getchar()) != '\n')
      putchar(ch); // dispose of bad input
    printf(" is not an integer.\n");
    printf("Enter an integer value,\n");
    printf("such as 25, -178, or 3: ");
  }
```

菜单浏览 V

```
  return input;
}
```

上机操作

例

编写一个程序，把输入作为字符流读取，直至遇到 EOF。令其报告输入中的大写字母个数和小写字母个数。

例

改写猜数程序：猜 1-100 中的某个数字 z，按二分法进行。

- 假设你最初猜 50 ，让其询问 z 是大于、小于还是等于猜测值。
- 若小于 50，则令下一次猜测值为 50 和 100 的平均值 75。
- 若大于 75，则令下一次猜测值为 50 和 75 的平均值 62。

例

编写一个程序，显示一个菜单，提供加法、减法、乘法或除法的选项。获取选择后，该程序请求两个数，然后执行选择的操作。

- 该程序应该只接受所提供的菜单选项，应使用 `float` 类型的数，并且如果用户未能输入数字应允许其重新输入。
- 在除法的情况下，如果用户输入 0 作为第二个数，该程序应该提示用户输入一个新的值。

```
Enter the operation of your choice:
a. add           b. substract
c. multiply      d. divide
q. quit
a
Enter first number: 22.4
Enter second number: one
one is not a number.
Please enter a number,
such as 2.5, -1.78E8, or 3: 1
22.40 + 1.00 = 23.40.
```

```
Enter the operation of your choice:
a. add          b. substract
c. multiply     d. divide
q. quit
d
Enter first number: 1
Enter second number: 0
Enter a number other than 0: 0.2
1.00 / 0.20 = 5.00.
Enter the operation of your choice:
a. add          b. substract
c. multiply     d. divide
q. quit
q
Bye.
```