# C语言综合研究与高强度程序设计训练17

# 1

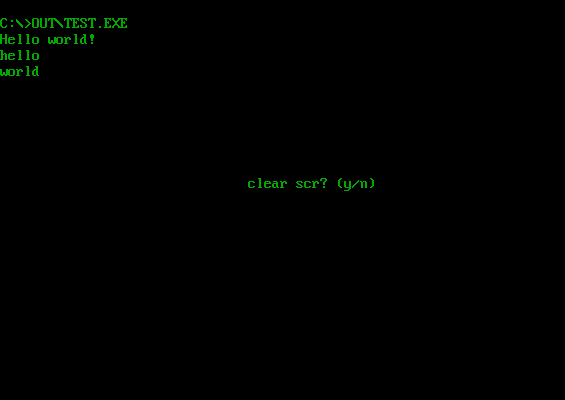
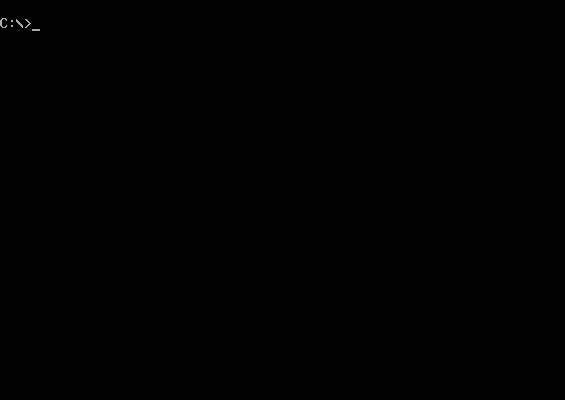
仔细研究fa函数，理解全部细节

fa实现的需求如下:

程序员可以调用这个函数提示用户是否进行相应的处理，然后接收用户的键盘回答  
果用户按下y键，返回非0，按下n键，返回0，为了提醒用户注意，提示信息用绿色的字符串在屏幕中间显示

* fa.c
* int fa(char \*str) {  
   int n; /\* 用来记录程序中得出字符串str的长度 \*/  
   int a;  
   int row; /\* 用来记录程序中计算出的显示字符串的起始行号 \*/  
   int col; /\* 用来记录程序中计算出的显示字串的起始列号 \*/  
   char far \*scr = (char far \*)0xb8000000;  
   char \*buf;  
   char ch;  
    
   /\* strlen函数的功能是求字符串的长度。 \*/  
   n = strlen(str);  
   /\* 根据字符串的长度计算显示字符串的行列位置 \*/  
   if (n / 80) {  
   row = (25 - (n / 80 + 1)) / 2;  
   col = 3;  
   } else {  
   row = 12;  
   col = (80 - (n % 80)) / 2;  
   }  
    
   /\* 保存原显存中相应空间中的内容。在屏幕中间显示提示字符串，有可能覆盖屏幕上原有的信息所以，在显示提示字符串前，将原来显存中相应空间里的内容保存起来，在函数返回前可以进行恢复，  
   \*/  
   buf = (char \*)malloc(n \* 2);  
    
   for (a = 0; a < n \* 2; a++) {  
   buf[a] = scr[(row - 1) \* 160 + (col - 1) \* 2 + a];  
   }  
    
   /\* 显示提示字符串。 \*/  
   for (a = 0; a < n; a++) {  
   scr[(row - 1) \* 160 + (col - 1 + a) \* 2] = str[a];  
   scr[(row - 1) \* 160 + (col - 1 + a) \* 2 + 1] = 2;  
   }  
    
   while (8) {  
   ch = getch();  
   if (ch == 'y' || ch == 'Y' || ch == 'n' || ch == 'N')  
   break;  
   }  
    
   /\* 收到用户的按键后，恢复显存相应空间中的内容 \*/  
   for (a = 0; a < n \* 2; a++) {  
   scr[(row - 1) \* 160 + (col - 1) \* 2 + a] = buf[a];  
   }  
    
   /\* 根据用户的按键，返回相应的状态信息 \*/  
   return (ch == 'y' || ch == 'Y');  
  }  
    
  main() {  
   int a;  
   char far \*p = (char far \*)0xb8000000;  
    
   printf("Hello world!\n");  
   printf("hello \n");  
   printf("world \n");  
    
   a = fa("use green paint? (y/n) ");  
    
   if (a)  
   for (a = 0; a < 2000; a++)  
   p[a \* 2 + 1] = 2;  
    
   a = fa("clear scr? (y/n)");  
    
   if (a)  
   system("cls");  
  }

## 2

* 以fa函数为基础，现一个函数，将ask函数的代码加入到c中，作为个库函数提供给程序员使用
* ask实现的需求如下:
* 程序员可以调用这个函数提示用户是否进行相应的处理，然后接收用户的键盘回答，如果用户按下y键，用传入的函数指针调用相应函数:按下键n，返回。为了提醒用户注意提示信息用绿色的字符串在屏幕中间显示
* ask函数的声明如下
* int ask(char\*,void(\*)())
* 下面的程序调用ask，提问用户是否将屏幕上的字符变为绿色，提问用户是否清屏，然后根据用户的回答进行相关的处理
* void f1();  
  void f2();  
    
  main() {  
   printf("Hello world!\n");  
   printf("hello \n");  
   printf("world \n");  
    
   fa("use green paint? (y/n) ", f1);  
    
   fa("clear scr? (y/n)", f2);  
  }  
    
  void f1() {  
   int a;  
   char far \*p = (char far \*)0xb8000000;  
   for (a = 0; a < 2000; a++)  
   p[a \* 2 + 1] = 2;  
  }  
  void f2() { system("cls"); }
* 修改后的ask.c
* int fa(char \*str, void (\*f)()) {  
   int n; /\* 用来记录程序中得出字符串str的长度 \*/  
   int a;  
   int row; /\* 用来记录程序中计算出的显示字符串的起始行号 \*/  
   int col; /\* 用来记录程序中计算出的显示字串的起始列号 \*/  
   char far \*scr = (char far \*)0xb8000000;  
   char \*buf;  
   char ch;  
    
   /\* strlen函数的功能是求字符串的长度。 \*/  
   n = strlen(str);  
   /\* 根据字符串的长度计算显示字符串的行列位置 \*/  
   if (n / 80) {  
   row = (25 - (n / 80 + 1)) / 2;  
   col = 3;  
   } else {  
   row = 12;  
   col = (80 - (n % 80)) / 2;  
   }  
    
   /\* 保存原显存中相应空间中的内容。在屏幕中间显示提示字符串，有可能覆盖屏幕上原有的信息所以，在显示提示字符串前，将原来显存中相应空间里的内容保存起来，在函数返回前可以进行恢复，  
   \*/  
   buf = (char \*)malloc(n \* 2);  
    
   for (a = 0; a < n \* 2; a++) {  
   buf[a] = scr[(row - 1) \* 160 + (col - 1) \* 2 + a];  
   }  
    
   /\* 显示提示字符串。 \*/  
   for (a = 0; a < n; a++) {  
   scr[(row - 1) \* 160 + (col - 1 + a) \* 2] = str[a];  
   scr[(row - 1) \* 160 + (col - 1 + a) \* 2 + 1] = 2;  
   }  
    
   while (8) {  
   ch = getch();  
   if (ch == 'y' || ch == 'Y' || ch == 'n' || ch == 'N')  
   break;  
   }  
    
   /\* 收到用户的按键后，恢复显存相应空间中的内容 \*/  
   for (a = 0; a < n \* 2; a++) {  
   scr[(row - 1) \* 160 + (col - 1) \* 2 + a] = buf[a];  
   }  
    
   /\* 根据用户的按键，返回相应的状态信息 \*/  
    
  + if (ch == 'y' || ch == 'Y') {  
  + f();  
  + }  
   return (ch == 'y' || ch == 'Y');  
  }
* 结果验证
* 
* 输入 y后
* 
* 输入y后
* 

## 3

* 思考几个问题:fa所实现的特性与ask所实现的特性有什么关系？既然有fa函数，ask函数的意义何在？ask函数的设计思想有什么普遍意义？
* 程序fa与ask都可以根据键入值来进行相关操作，但是ask可以通过函数指针来执行各种各样个操作但就ask的代码复杂度来说没有太大的变化反观fa就要进行各种各样的个性操作。
* 通过ask可以使代码更加集成，让程序员可以关注宏观上的实现。对每一个相应的功能可以按照自己的想法来更改不必每一次改动一个小功能都要修改ask
* 通过ask的设计思想就类似模板，放入不同的功能执行出来的效果也不相同。