云南大学数学与统计学实验教学中心  
《高级语言程序设计》实验报告

|  |  |  |
| --- | --- | --- |
| 课程名称：程序设计和算法语言 | 学期：2016~2017学年上学期 | 成绩： |
| 指导教师：赵越 | 学生姓名：刘鹏 | 学生学号：20151910042 |
| 实验名称：模块化程序设计 |  |  |
| 实验编号：No.06 | 实验日期：2018年8月19日 | 实验学时：2 |
| 学院：数学与统计学院 | 专业：信息与计算科学 | 年级：2015级 |

# 实验目的

掌握C语言中定义函数的方法及其调用方法。

掌握函数实参与形参的对应关系以及“值传递”与“地址传递”的方式。

掌握全局变量和局部变量、动态变量与静态变量的概念和使用方法。

掌握函数的嵌套调用与递归调用的方法。

掌握通过数组进行数据传递的方法。

学会使用宏替换编写程序，弄清“文件包含”的作用。

学会全局变量和局部变量、动态变量和静态变量的概念和使用方法。

# 实验环境

Windows10 Pro Workstation 17134.288；

Cygwin GCC编译器。

# 实验内容

1. 变量的生存期是一个静态的概念，是不变的，作用域是一个动态的概念，随程序的运行而变。从作用域角度分，有**局部变量**和**全局变量**；从变量存在的时间来分，有**动态存贮**和**静态存贮**。

2. C语言中的参数调用为值调用，名调用（地址）通过指针实现的。

3. 用递归方式编写程序，一般由三部分组成：

递归条件；

递归化简；

递归出口；

一个递归总是可分为“回推”和“递推”两个阶段，即从要解决问题出发，一步步推到已知条件，然后再由已知条件一步步找到结论。

## 函数调用练习

通过运行下面程序，熟悉函数的调用方法。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | /\*  \* filename: 6.1 test.c  \* property: test  \*/  #include <stdio.h>  void fun**(**int i**,** int j**,** int k**)** **{**  int t**;**  t**=(**i **+** j **+** k**)** **/** 2**;**  printf**(**"t = %d\n"**,** t**);**  **}**  int main**()** **{**  int x**,** y**,** z**;**  x **=** 4**;**  y **=** 12**;**  z **=** 6**;**  fun**(**x**,** y**,** z**);**  printf**(**"x = %d; y = %d; z = %d\n"**,** x**,** y**,** z**);**  **return** 0**;**  **}** |

### 运行结果



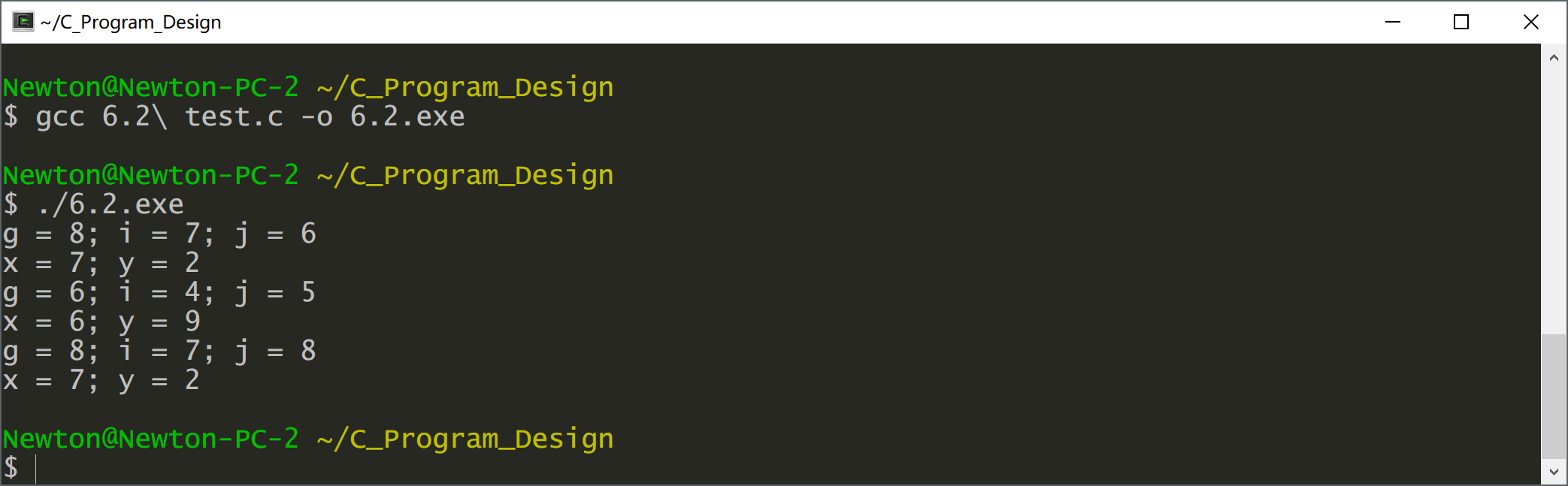
## 程序调用

运行下面程序，写出执行结果。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | /\*  \* filename: 6.2 test.c  \* property: test  \*/  #include <stdio.h>  void f**(**int i**,**int j**)** **{**  int x**,** y**,** g**;**  g **=** 8**;**  x **=** 7**;**  y **=** 2**;**  printf**(**"g = %d; i = %d; j = %d\n"**,** g**,** i**,** j**);**  printf**(**"x = %d; y = %d\n"**,** x**,** y**);**  **}**  int main**()** **{**  int i**,** j**,** x**,** y**,** n**,** g**;**  i **=** 4**;**  j **=** 5**;**  g **=** x **=** 6**;**  y **=** 9**;**  n **=** 7**;**  f**(**n**,** 6**);**  printf**(**"g = %d; i = %d; j = %d\n"**,** g**,** i**,** j**);**  printf**(**"x = %d; y = %d\n"**,** x**,** y**);**  f**(**n**,** 8**);**  **return** 0**;**  **}** |

### 运行结果



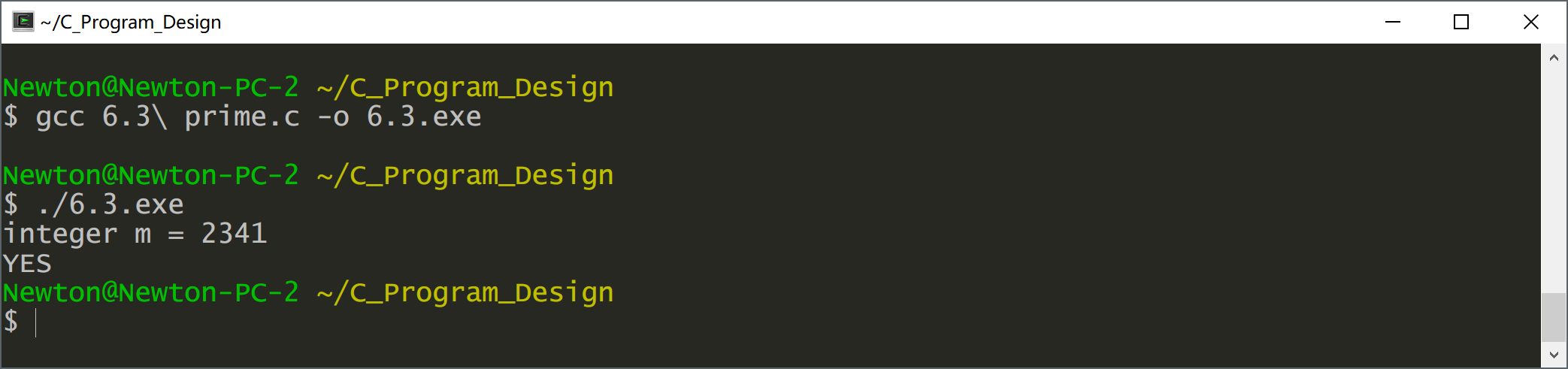
## 素数判断

编写一个判断素数的函数，在主函数输入一个整数，输出是否是素数的信息。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | /\*  \* filename: 6.3 prime.c  \* property: exercise  \*/  #include <stdio.h>  #include <math.h>  void Prime**(**int n**)** **{**  int i**,** k**;**  k **=** n**/**2**;**  **for(**i **=** 2**;** i **<=** k**;** i**++)**  **if** **(**n **%** i **==** 0**)**  **break;**  **if(**i **>=** k **+** 1**)** **{**  printf**(**"YES"**);**  **}**  **else** **{**  printf**(**"NO"**);**  **}**  **}**  int main**()** **{**  int m**;**  printf**(**"integer m = "**);**  scanf**(**"%d"**,&**m**);**  Prime**(**m**);**  **return** 0**;**  **}** |

### 运行结果



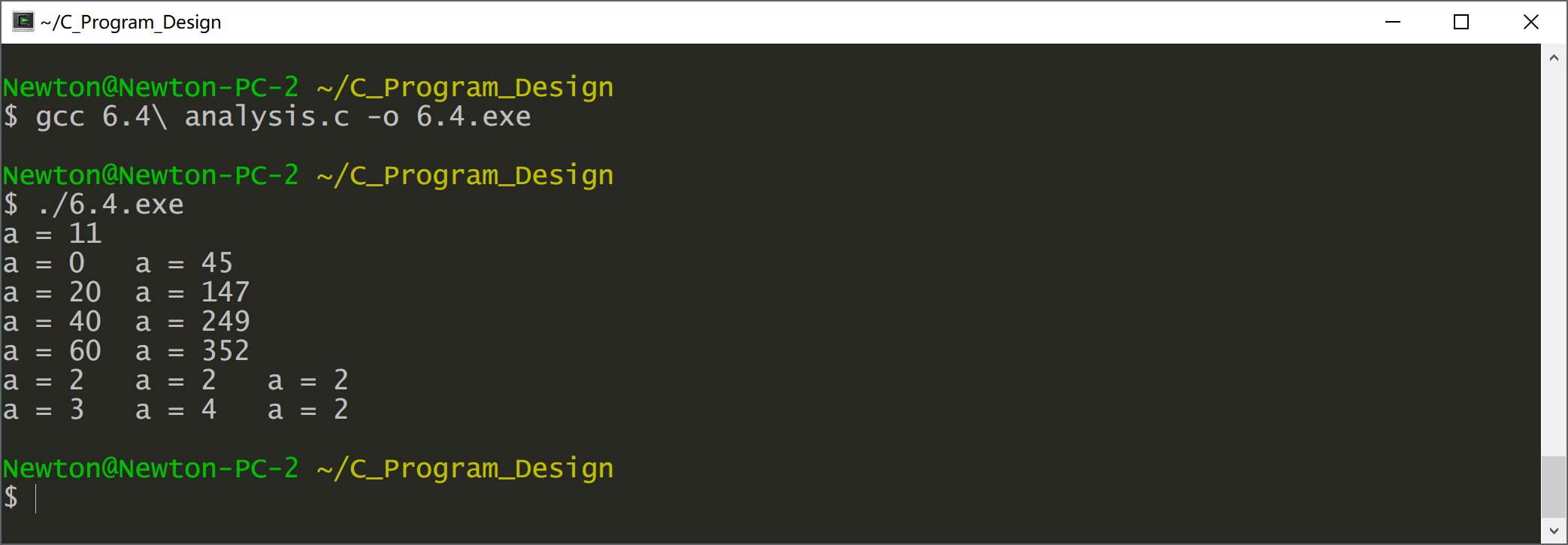
## 代码分析

先读懂程序，分析出结果，然后上机运行此程序。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | /\*  \* filename: 6.4 analysis.c  \* property: analysis  \*/  #include <stdio.h>  #define FUE(K) K + 3.14159  #define PR(a) printf("a = %d\t", (int)(a))  #define PRINT(a) PR(a); putchar('\n')  #define PRINT2(a, b) PR(a); PRINT(b)  #define PRINT3(a, b, c) PR(a); PRINT2(b,c)  #define MAX(a, b) (a < b? b : a)  int main**()** **{**  **{**  int x **=** 2**;**  PRINT**(**x **\*** FUE**(**4**));**  **}**    **{**  int f**;**  **for(**f **=** 0**;** f **<=** 60**;** f **+=** 20**)** **{**  PRINT2**(**f**,** 5.12 **\*** f **+** 45**);**  **}**  **}**    **{**  int x **=** 1**,** y **=** 2**;**  PRINT3**(**MAX**(**x**++,** y**),** x**,** y**);**  PRINT3**(**MAX**(**x**++,** y**),** x**,** y**);**  **}**  **return** 0**;**  **}** |

### 运行结果



可以看出，当进行了宏定义之后，不加括号与加括号是有很大区别的。

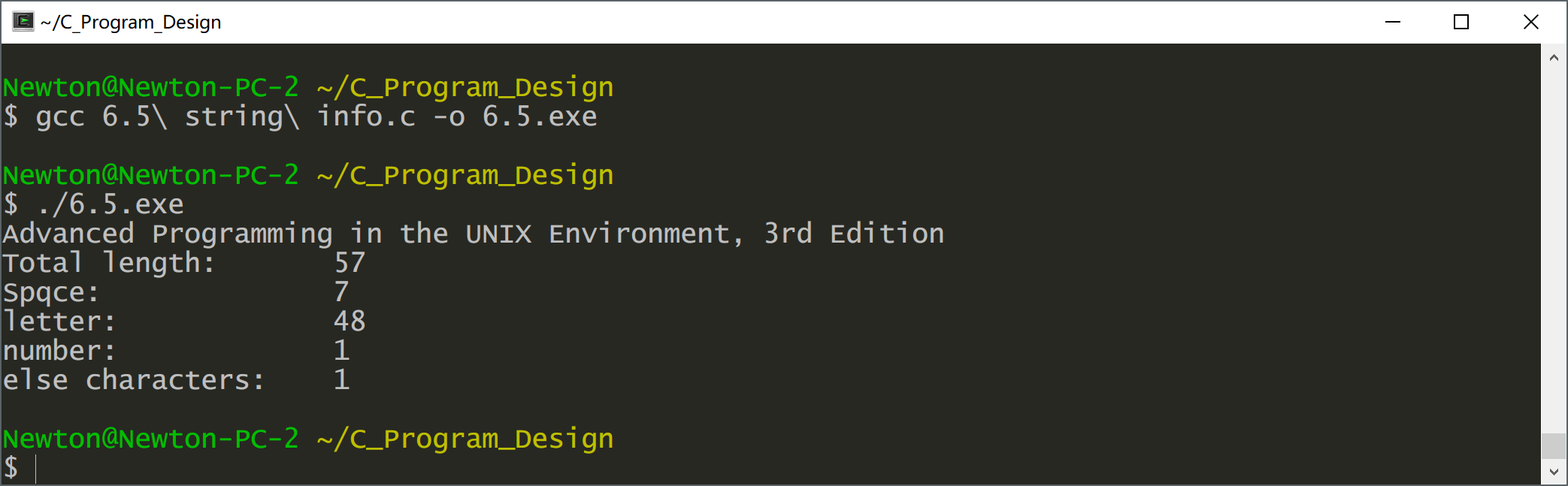
## \*统计字符串的字符信息

编写一函数，由实参传来一个字符串，统计此字符串中字母、数字、空格和其它字符的个数，在主函数中输入字符串，输出上述结果。此程序编写时最好把存放字母、数字、空格和其它字符的变量定义为全程变量，这样在函数中就不必定义了。用其它方法也可以。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | /\*  \* filename: 6.5 string info.c  \* property: exercise  \*/  #include <string.h>  #include <stdio.h>  int main**()** **{**  int space **=** 0**;**  int alpha **=** 0**;**  int numbs **=** 0**;**  int elsec **=** 0**;**  char a**[**1000**];**  gets**(**a**);**  int i**;**  **for** **(**i **=** 0**;** i **<=** strlen**(**a**);** i**++)** **{**  **if** **(**a**[**i**]** **>=** 'a' **&&** a**[**i**]** **<=** 'z' **||** a**[**i**]** **>=** 'A' **&&** a**[**i**]** **<=** 'Z'**)** **{**  alpha **+=** 1**;**  **}**  **if** **(**a**[**i**]** **==** ' '**)** **{**  space **+=** 1**;**  **}**  **if** **(**a**[**i**]** **>=** '0' **&&** a**[**i**]** **<=** '9'**)** **{**  numbs **+=** 1**;**  **}**  **}**  elsec **=** strlen**(**a**)** **-** space **-** alpha **-** numbs**;**  printf**(**"Total length: %d\n"**,** strlen**(**a**));**  printf**(**"Spqce: %d\n"**,** space**);**  printf**(**"letter: %d\n"**,** alpha**);**  printf**(**"number: %d\n"**,** numbs**);**  printf**(**"else characters: %d\n"**,** elsec**);**  **return** 0**;**  **}** |

### 运行结果



## 递归函数编写

写出计算Ackermann函数Ack(m,n)的递归函数，对于m≥0,n≥0,Ack(m,n)定义为：

对于，，要显示出计算结果。编程并上机调试。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | /\*  \* filename: 6.6 recursion.c  \* property: exercise  \*/  #include <stdio.h>  int ACK**(**int m**,** int n**)** **{**  **if(**m **==** 0 **&&** n **>=** 0**)**  **return** 1 **+** n**;**  **else**  **if(**n **==** 0 **&&** m **>** 0**)**  **return** ACK**(**m **-** 1**,** 0**);**  **else**  **if(**m **>** 0 **&&** n **>** 0**)**  **return** ACK**(**m **-** 1**,** ACK**(**m**,** n **-** 1**));**  **}**  int main**()** **{**  int m**,** n**;**  scanf**(**"%d %d"**,** **&**m**,** **&**n**);**  printf**(**"ACK(%d, %d) = %d\n"**,**m**,** n**,** ACK**(**m**,** n**));**  getchar**();**  **}** |

### 运行结果



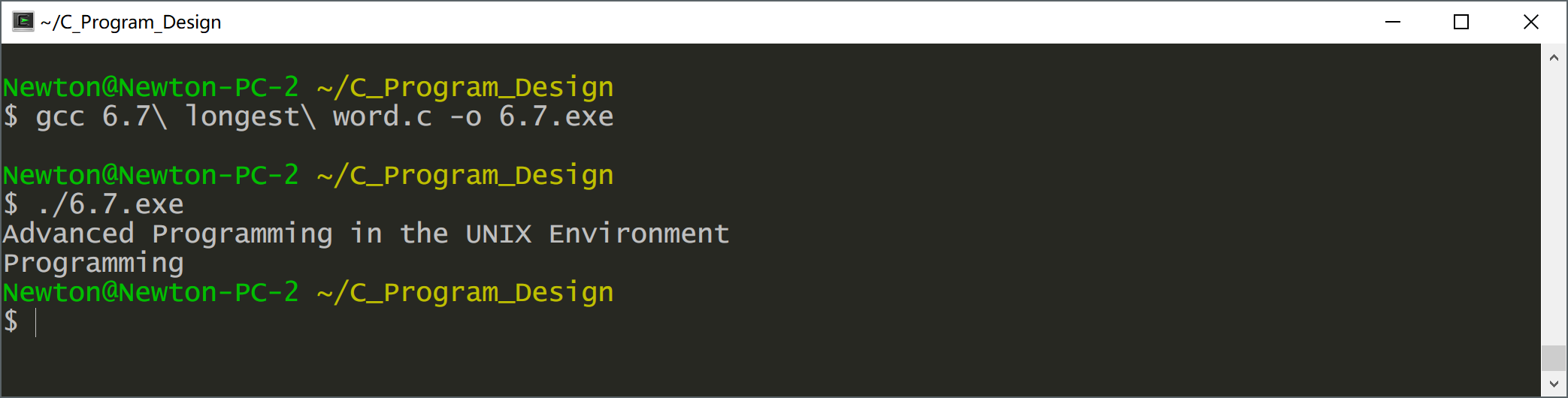
## \*最长单词

写一函数，输入一行字符，将此字符串中最长的单词输出。编写程序并上机调试运行，记录下程序运行的结果。提示：我们认为每行字符都是由字母组成的字符串，在程序中设函数为实现寻找最长的字符串。设标记flag表示单词是否开始，flag=1表示单词开始，flag=0表示单词未开始。Point代表当前单词的起始位置，place代表最长单词的起始位置。Len代表当前单词已累计字母的个数，length代表先前单词中最长单词的长度。下面给一个程序段，上实现寻找最长的单词。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | /\*  \* filename: 6.7 longest word.c  \* property: exercise  \*/  #include <stdio.h>  #include <string.h>  int is\_letter**(**char c**)** **{**  **if** **(**c **>=** 'a' **&&** c **<=** 'z' **||** c **>=** 'A' **&&** c **<=** 'Z'**)** **{**  **return** 1**;**  **}**  **else** **{**  **return** 0**;**  **}**  **}**  int main**()** **{**  char a**[**5000**];**  gets**(**a**);**  int max **=** 0**;**  int max\_position **=** 0**;**  int word\_length **=** 0**;**  int i**;**  **for** **(**i **=** 0**;** i **<=** strlen**(**a**);** i**++)** **{**  **if** **(**is\_letter**(**a**[**i**])** **==** 1**)** **{**  word\_length **+=** 1**;**  **}**  **else** **{**  **if** **(**word\_length **>** max**)** **{**  max **=** word\_length**;**  max\_position **=** i **-** max**;**  **}**  word\_length **=** 0**;**  **}**  **}**  **for** **(**i **=** max\_position**;** i **<** max\_position **+** max**;** i**++)** **{**  printf**(**"%c"**,** a**[**i**]);**  **}**  **return** 0**;**  **}** |

### 运行结果



## 最大公因数与最小公倍数

求两整数的最大公约数和最小公倍数。用一函数求最大公约数，用另一函数调用此函数求出最大公约数，并用求出的最大公约数求最小公倍数。

具体要求如下：

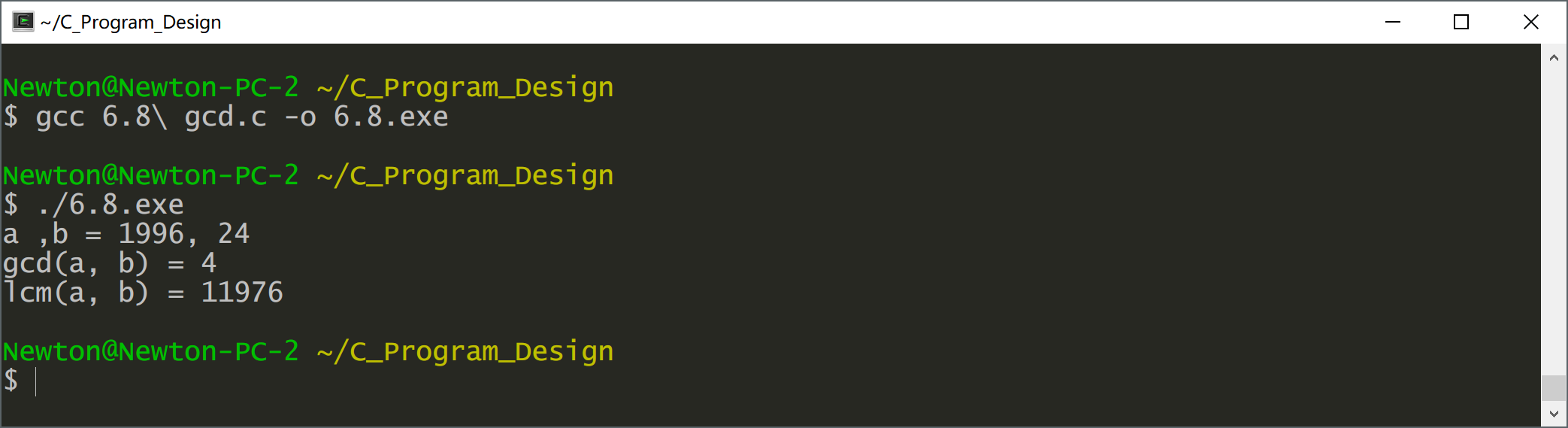
①用全局变量。将最大公约数与最小公倍数设为全局变量，在主函数中输出它们的值。

②不用全局变量。最大公约数和最小公倍数由被调模块返回值。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | /\*  \* filename: 6.8 gcd.c  \* property: exercise  \*/  #include <stdio.h>  int gcd**(**int a**,** int b**)** **{**  **if** **(**a **==** 0 **||** b **==** 0**)** **{**  **return** a **+** b**;**  **}**  **else** **{**  **if** **(**a **>** b**)** **{**  **return** gcd**(**a **-** b**,** b**);**  **}**  **else** **{**  **return** gcd**(**a**,** b **-** a**);**  **}**  **}**  **}**  int main**()** **{**  int a**,** b**;**  printf**(**"a ,b = "**);**  scanf**(**"%d, %d"**,** **&**a**,** **&**b**);**  printf**(**"gcd(a, b) = %d\n"**,** gcd**(**a**,** b**));**  printf**(**"lcm(a, b) = %d\n"**,** a **\*** b **/** gcd**(**a**,** b**));**  **return** 0**;**  **}** |

### 运行结果



## 组合数计算

计算并输出

具体要求如下：

①编制一个函数，返回值。

②编制主函数，由键盘输入与，调用计算下列算式值。

③在主函数中，输入与之前要有提示，并检查输入数据的合理性，对于不合理的输入，应输出出错信息，不再进行计算。在函数中也要检查的合理性，当时输出出错信息，不再进行计算。

④分别输入，，，，运行该程序。

⑤画出模块的流程图。

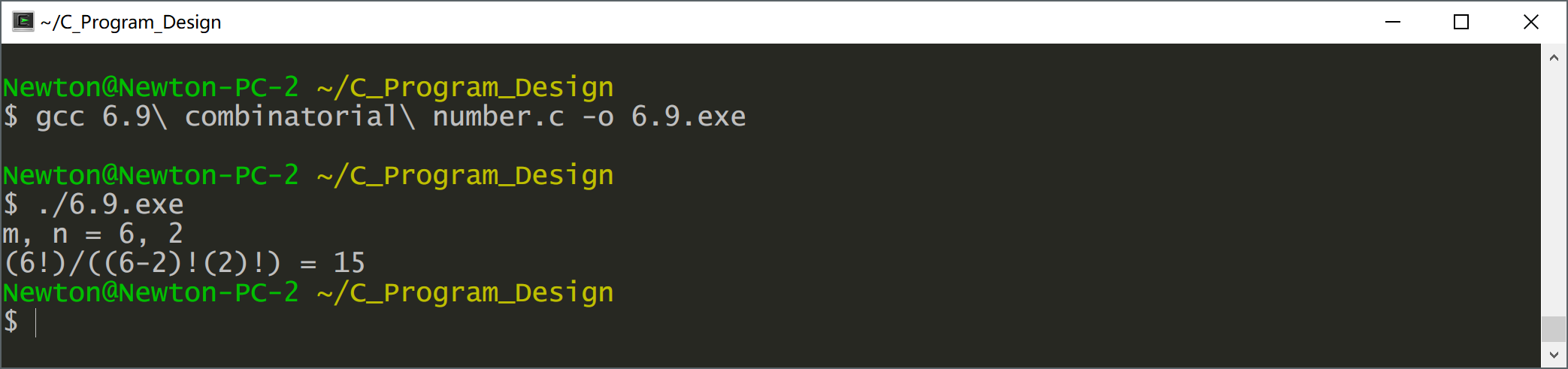
### 流程图



### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | /\*  \* filename: 6.9 combinatorial number.c  \* property: exercise  \*/  #include <stdio.h>  #define INPUT\_ERROR -1  int factorial**(**int a**)** **{**  **if** **(**a **<** 0**)** **{**  **return** INPUT\_ERROR**;**  **}**  **if** **(**a **==** 0**)** **{**  **return** 1**;**  **}**  **if** **(**a **>** 1**)** **{**  **return** a **\*** factorial**(**a **-** 1**);**  **}**  **}**  int main**()** **{**  int m**,** n**;**  printf**(**"m, n = "**);**  scanf**(**"%d, %d"**,** **&**m**,** **&**n**);**  **if** **(**m **\*** n **<=** 0 **||** m **<** n**)** **{**  **return** INPUT\_ERROR**;**  **}**  **else** **{**  int t **=** factorial**(**m**)** **/** **(**factorial**(**n**)** **\*** factorial**(**m **-** n**));**  printf**(**"(%d!)/((%d-%d)!(%d)!) = %d"**,** m**,** m**,** n**,** n**,** t**);**  **}**  **return** 0**;**  **}** |

### 运行结果



## 判别

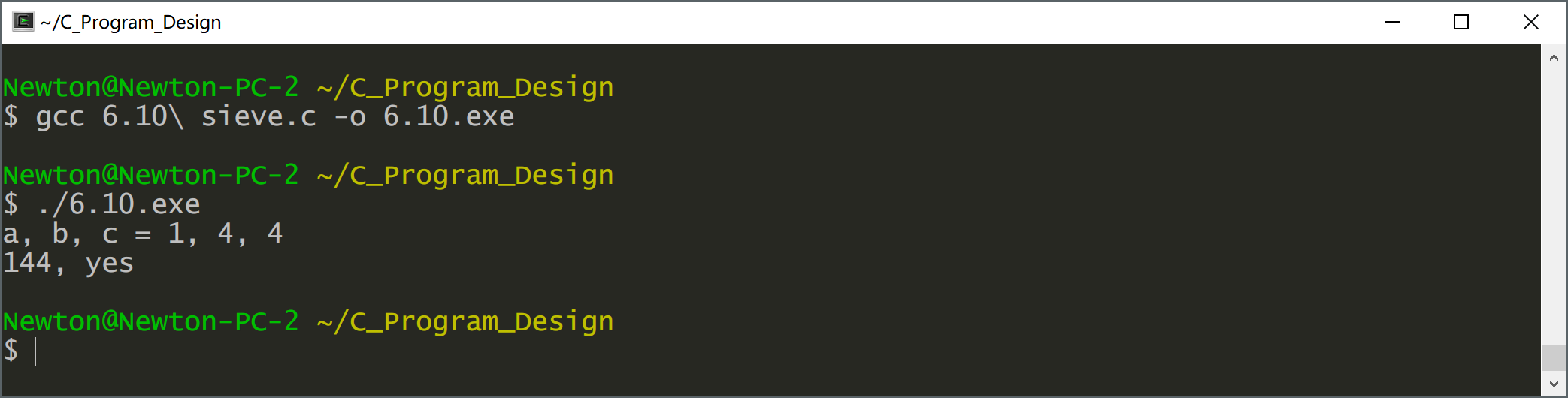
编写程序，要求找出满足下列条件的3位数：它是完全平方数，又有两位数字相同。如：144、676。

要求：设计一函数判断一个三位数是否为完全平方数，设计另一函数判断一个三位数中是否有两位数字相同，再在主函数中调用这两个函数，找出所有的满足这两个条件的三位数。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | /\*  \* filename: 6.10 sieve.c  \* property: exercise  \*/  #include <stdio.h>  #include <math.h>  int judge**(**int a**,** int b**,** int c**)** **{**  int BOOL**;**  float z**;**  float k**;**  float j**;**  z **=** 100 **\*** a **+** 10 **\*** b **+** c**;**  k **=** sqrt**(**z**);**  **for(**j **=** 1**;** j **<** z**;** j**++)** **{**  **if(**j **-** k **==** 0**)**  BOOL **=** 1**;**  **continue;**  **}**  **return** BOOL**;**  **}**  int compare**(**int a**,** int b**,** int c**)** **{**  int BOOL**;**  **if((**a **==** b **&&** a **!=** c**)** **||** **(**a **==** c **&&** a **!=** b**)** **||** **(**b **==** c **&&** a **!=** b**))** **{**  BOOL **=** 1**;**  **}**  **return** BOOL**;**  **}**  int main**()** **{**  int a**,** b**,** c**;**  printf**(**"a, b, c = "**);**  scanf**(**"%d, %d, %d"**,** **&**a**,** **&**b**,** **&**c**);**  **if(**judge**(**a**,** b**,** c**)** **==** 1 **&&** compare**(**a**,** b**,** c**)** **==** 1**)** **{**  printf**(**"%d%d%d, yes\n"**,** a**,** b**,** c**);**  **}**  **else{**  printf**(**"Sorry, it isn't."**);**  **}**  **}** |

### 运行结果



编写函数，将个整数的数列进行重新排放，重新排放后的结果为：前段都是奇数，后段都是偶数，并编写主函数完成：

①输入10个整数；

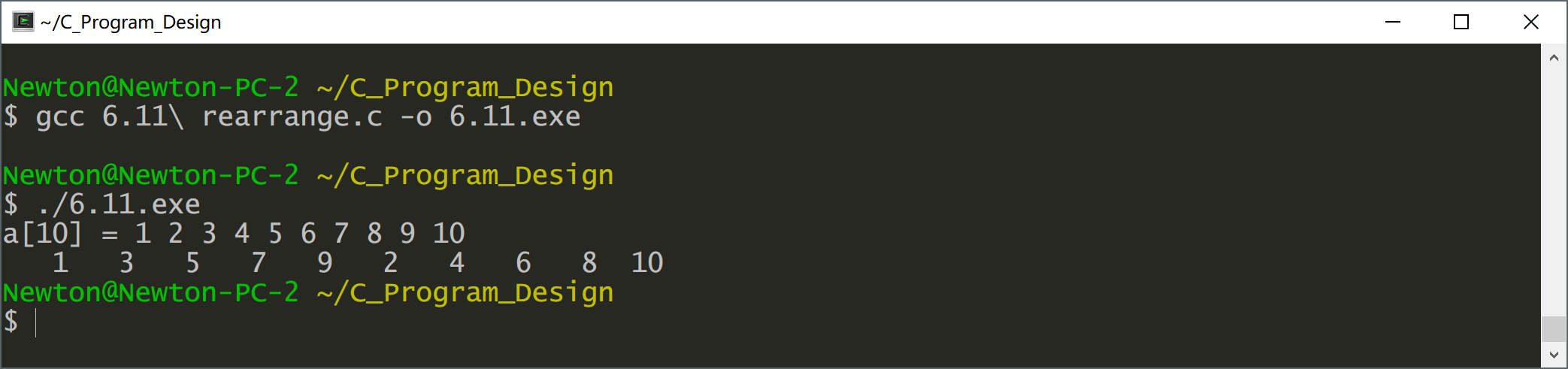
②调用此函数进行重排；

③输出重排后的结果。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | /\*  \* filename: 6.11 rearrange.c  \* property: exercise  \*/  #include <stdio.h>  void arrange**(**int a**[])** **{**  int count **=** 0**;**  int i**;**  int ans**[**10**];**  **for** **(**i **=** 0**;** i **<** 10**;** i**++)** **{**  **if** **(**a**[**i**]** **%** 2 **==** 1**)** **{**  ans**[**count**]** **=** a**[**i**];**  count **+=** 1**;**  **}**  **}**  **for** **(**i **=** 0**;** i **<** 10**;** i**++)** **{**  **if** **(**a**[**i**]** **%** 2 **==** 0**)** **{**  ans**[**count**]** **=** a**[**i**];**  count **+=** 1**;**  **}**  **}**  **for** **(**i **=** 0**;** i **<** 10**;** i**++)** **{**  a**[**i**]** **=** ans**[**i**];**  **}**  **}**  int main**()** **{**  int i**;**  int a**[**10**];**  printf**(**"a[10] = "**);**  **for** **(**i **=** 0**;** i **<** 10**;** i**++)** **{**  scanf**(**"%d"**,** **&**a**[**i**]);**  **}**    arrange**(**a**);**  **for** **(**i **=** 0**;** i **<** 10**;** i**++)** **{**  printf**(**"%4d"**,** a**[**i**]);**  **}**  **return** 0**;**  **}** |

### 运行结果



## 学生成绩统计

输入10个学生4门课的成绩，分别用函数求：

①每个学生的平均成绩；

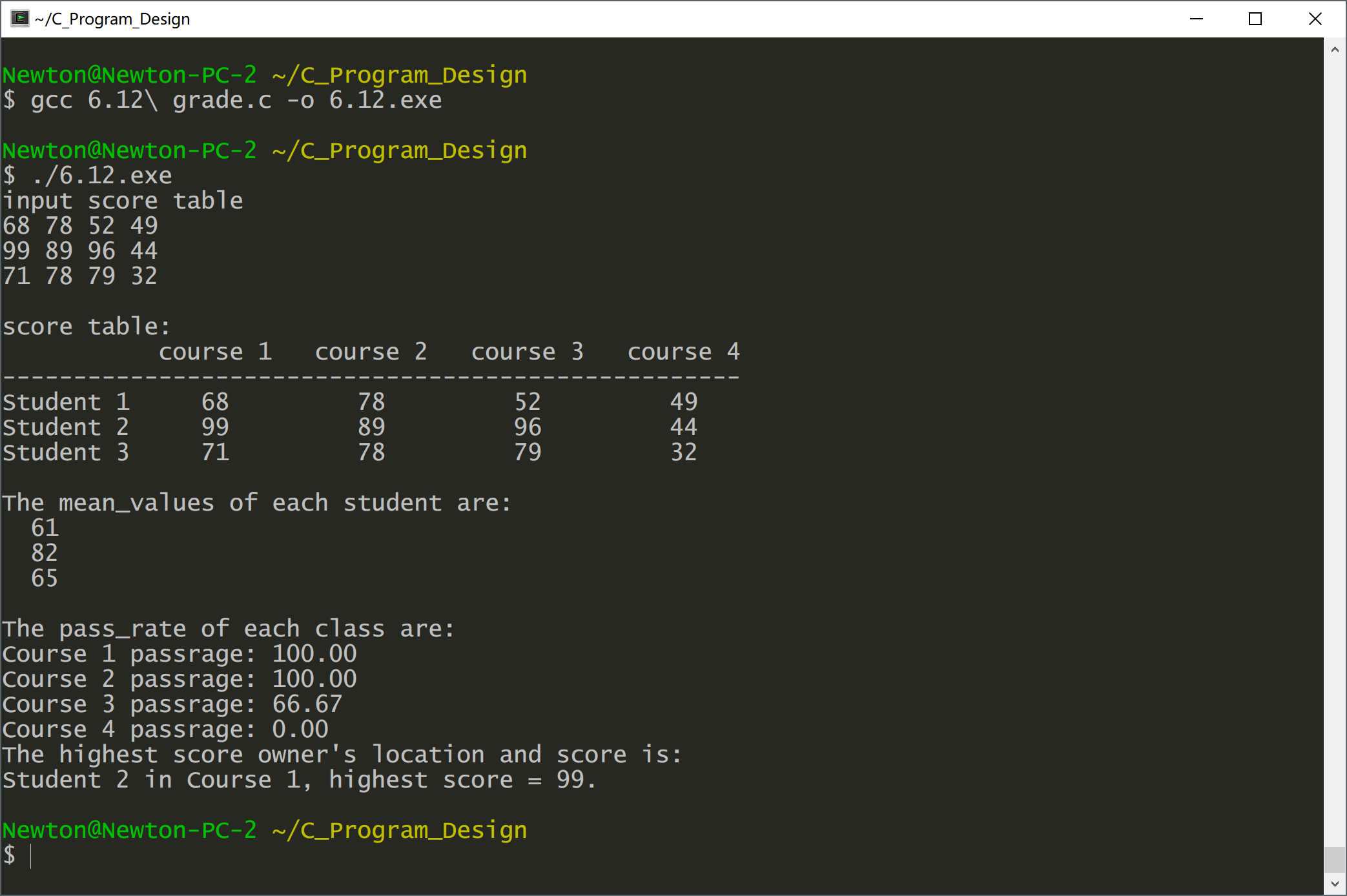
②每门课的及格率；

③最高分所对应的学生和课程。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102 | /\*  \* filename: 6.12 grade.c  \* property: exercise  \*/  #include <stdio.h>  #define ROW 3  #define COLUMN 4  void mean**(**int a**[**ROW**][**COLUMN**])** **{**  int sum**[**ROW**];**  int avg**[**ROW**];**  int i**,** j**;**  **for(**j **=** 0**;** j **<** ROW**;** j**++)** **{**  sum**[**j**]** **=** 0**;**  **for(**i **=** 0**;** i **<** COLUMN**;** i**++)** **{**  sum**[**j**]** **+=** a**[**j**][**i**];**  **}**  avg**[**j**]** **=** sum**[**j**]** **/** COLUMN**;**  **}**  **for(**j **=** 0**;** j **<** ROW**;** j**++)** **{**  printf**(**"%4d\n"**,** avg**[**j**]);**  **}**  printf**(**"\n"**);**  **}**  void passrate**(**int a**[**ROW**][**COLUMN**])** **{**  float count**[**COLUMN**];**  int i**,** j**;**  **for(**j **=** 0**;** j **<** COLUMN**;** j**++)** **{**  **for(**i **=** 0**;** i **<** ROW**;** i**++)** **{**  **if(**a**[**i**][**j**]** **>=** 60**)** **{**  count**[**j**]++;**  **}**  **}**  **}**  **for(**i **=** 0**;** i **<** COLUMN**;** i**++)** **{**  printf**(**"Course %d passrage: %2.2f\n"**,** i **+** 1**,** **(**count**[**i**]** **/** ROW**)** **\*** 100.0**);**  **}**  **}**  int max2**(**int a**,**int b**)** **{**  **return(**a **>** b**?** a **:** b**);**  **}**  void max**(**int a**[**ROW**][**COLUMN**])** **{**  int Max**;**  Max **=** a**[**0**][**0**];**  int column**,** row**;**  int i**,** j**;**  **for(**i **=** 0**;** i **<** ROW**;** i**++)** **{**  **for(**j **=** 0**;** j **<** COLUMN**;** j**++)** **{**  **if(**Max **<** a**[**i**][**j**])** **{**  Max **=** a**[**i**][**j**];**  row **=** i**;**  column **=** j**;**  **}**  **}**  **}**  printf**(**"Student %d in Course %d, highest score = %d.\n"**,** row **+** 1**,** column **+** 1**,** Max**);**  **}**  int main**()** **{**  int i**,** j**;**  int a**[**ROW**][**COLUMN**];**  printf**(**"input score table\n"**);**  **for** **(**i **=** 0**;** i **<** ROW**;** i**++)** **{**  **for** **(**j **=** 0**;** j **<** COLUMN**;** j**++)** **{**  scanf**(**"%d"**,** **&**a**[**i**][**j**]);**  **}**  **}**  printf**(**"\nscore table:\n"**);**  char space**[]** **=** " "**;**  printf**(**"%s"**,** space**);**  **for** **(**i **=** 1**;** i **<=** COLUMN**;** i**++)** **{**  printf**(**"course %d "**,** i**);**  **}**  printf**(**"\n"**);**  **for** **(**i **=** 0**;** i **<** 8 **\*** COLUMN **+** 11 **+** 3 **\*** **(**COLUMN **-** 1**);** i**++)** **{**  printf**(**"-"**);**  **}**  printf**(**"\n"**);**  **for** **(**i **=** 0**;** i **<** ROW**;** i**++)** **{**  printf**(**"Student %d "**,** i **+** 1**);**  **for(**j **=** 0**;** j **<** COLUMN**;** j**++)** **{**  printf**(**"%d "**,** a**[**i**][**j**]);**  **if(**j **==** COLUMN**-**1**)** **{**  printf**(**"\n"**);**  **}**  **}**  **}**  printf**(**"\nThe mean\_values of each student are:\n"**);**  mean**(**a**);**  printf**(**"The pass\_rate of each class are:\n"**);**  passrate**(**a**);**  printf**(**"The highest score owner's location and score is:\n"**);**  max**(**a**);**  **return** 0**;**  **}** |

### 运行结果



## 习题

1．下列说法哪些是不正确的

1）C语言的函数必须在主函数的前面定义。（×）

2）函数名前面都必须写数据类型和存贮类型。（×）

3）一个C程序可以包含多个函数，并且必须有最多有一个主函数。（√）

4）函数的外部不允许再说明其它变量。（×）

5）void类函数不能有返回值。（√）

6）函数不一定都有返回值，不管其类型是否为void。（×）

7）引用程序中的函数被调用函数必须在调用函数前面预先定义。（√）

8）在函数内部定义的变量都是局部量。（√）

9）所有外部量，都必须在其它程序中定义过。

10）函数返回的数据类型必须和函数数据类型一致。（√）

11）调用函数的实参与被调用函数的形参必须保证其个数、次序类型完全一致。（√）

12）在C程序中，函数嵌套调用不允许超过18层。（×）

## 程序排错

下列定义有何错误？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50 | /\*----------------------------------\*/  float f**(**x**,**y**);**  int x**,**y**;**  **{**  **}**  /\*----------------------------------\*/  f**(**int x**,**y**)**  **{**  retuen x**\***y**;**  **}**  /\*----------------------------------\*/  /\*----------------------------------\*/  int max**(**x**,**y**)** /\*定义求最大值的函数\*/  **{** int x**,**y**,**z**;**  z**=**x**>**y**?** x**:**y**;**  **return** z**;**  **}**  （4）  int f**(**x**,**y**,**z**)**  int x**,**y**;**  **{**  int z**;**  **:**  **}**  /\*----------------------------------\*/  int f**(**int n**)** /\*定义求n!的函数\*/  **{**  **if** n**<=**0 **return** 1**;**  **else** f**=**n**\***f**(**n**-**1**);**  **}**  /\*----------------------------------\*/  int main**(**x**,**y**)**  int x**,**y**;**  **{**  printf**(**“x**+**y**=%**d\n”**,**x**+**y**);**  **}**  int x**,**y**;**  f**(**int x**)**  **{**  y**=**x**\***x**;** **return** y **;}**  int f1**(**x**)**  int x**;**  **{**  int f2**(**y**)**  int y**;**  **{** …  **}**  **}** |

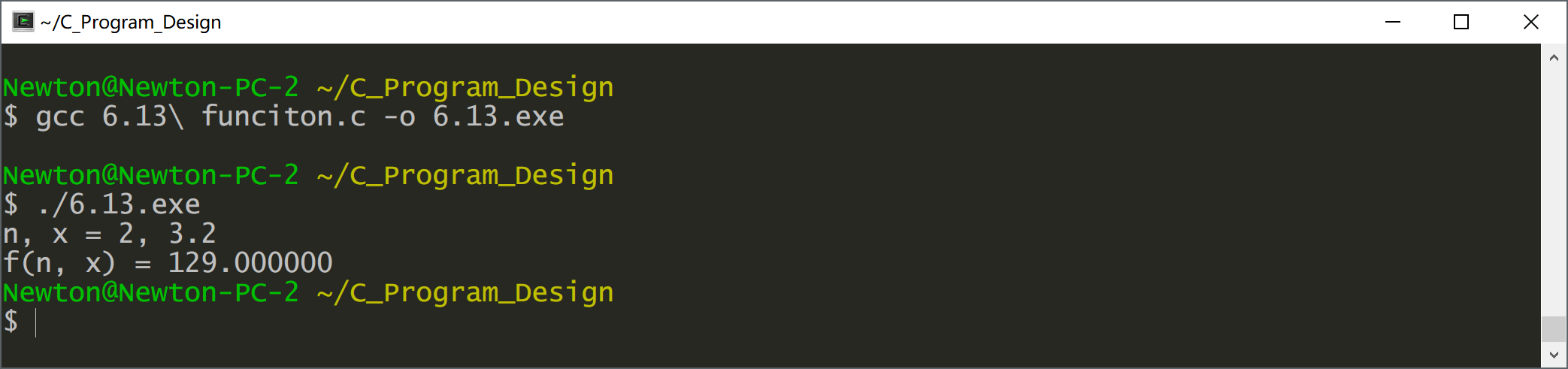
## 多态函数

定义一个函数，调用程序通过f(n,x)的形式就可能计算，，等样式的表达式的值。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | /\*  \* filename: 6.13 function.c  \* property: exercise  \*/  #include <stdio.h>  #include <math.h>  void f**(**int n**,** double x**)** **{**  **switch(**n**)** **{**  **case** 1**:** printf**(**"%f"**,** pow**(**x**,** 3**)** **+** x **-** 1**);** **break;**  **case** 2**:** printf**(**"%f"**,** pow**((**x **+** 5**),** 3**)** **+** **(**x **+** 5**)** **-** 1**);** **break;**  **case** 3**:** printf**(**"%f"**,** pow**(**sin**(**x**),** 3**)** **+** sin**(**x**)** **-** 1**);** **break;**  **default:** printf**(**"Error input"**);**  **}**  **}**  int main**()** **{**  printf**(**"n, x = "**);**  int n**;**  double x**;**  scanf**(**"%d, %f"**,** **&**n**,** **&**x**);**  printf**(**"f(n, x) = "**);**  f**(**n**,** x**);**  **return** 0**;**  **}** |

### 运行结果



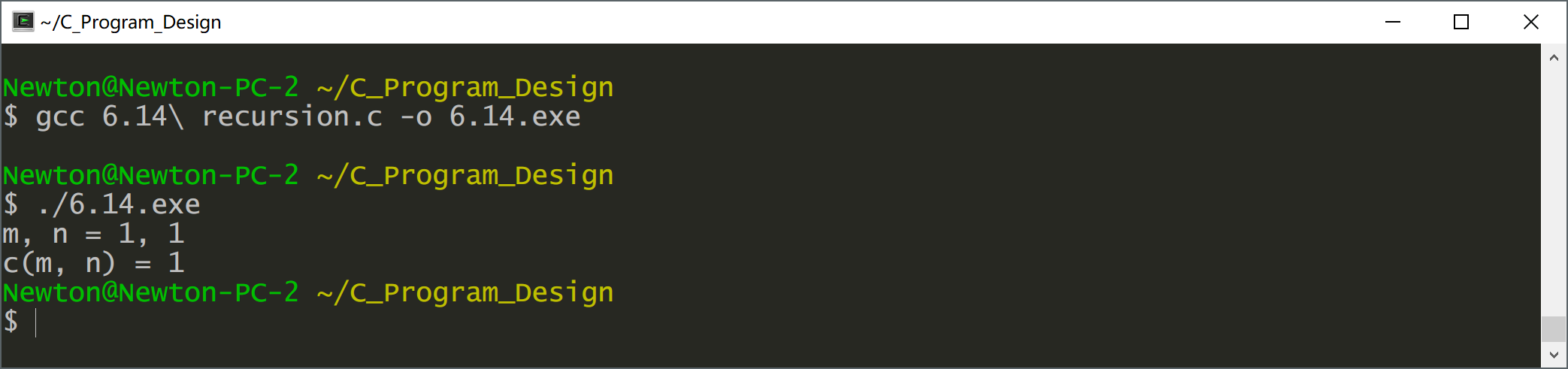
## 递归函数

写一递归函数，计算

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | /\*  \* filename: 6.14 recursion.c  \* property: exercise  \*/  #include <stdio.h>  int c**(**int m**,** int n**)** **{**  **if** **(**n **==** 0 **||** m **==** n**)** **{**  **return** 1**;**  **}**  **if** **(**n **==** 1**)** **{**  **return** m**;**  **}**  **if** **(**m **>** n **>** 1**)** **{**  **return** c**(**m **-** 1**,** n **-** 1**)** **+** c**(**n**,** m **-** 1**);**  **}**  **}**  int main**()** **{**  int m**,** n**;**  printf**(**"m, n = "**);**  scanf**(**"%d, %d"**,** **&**m**,** **&**n**);**  printf**(**"c(m, n) = %d"**,** c**(**m**,** n**));**  **}** |

### 运行结果



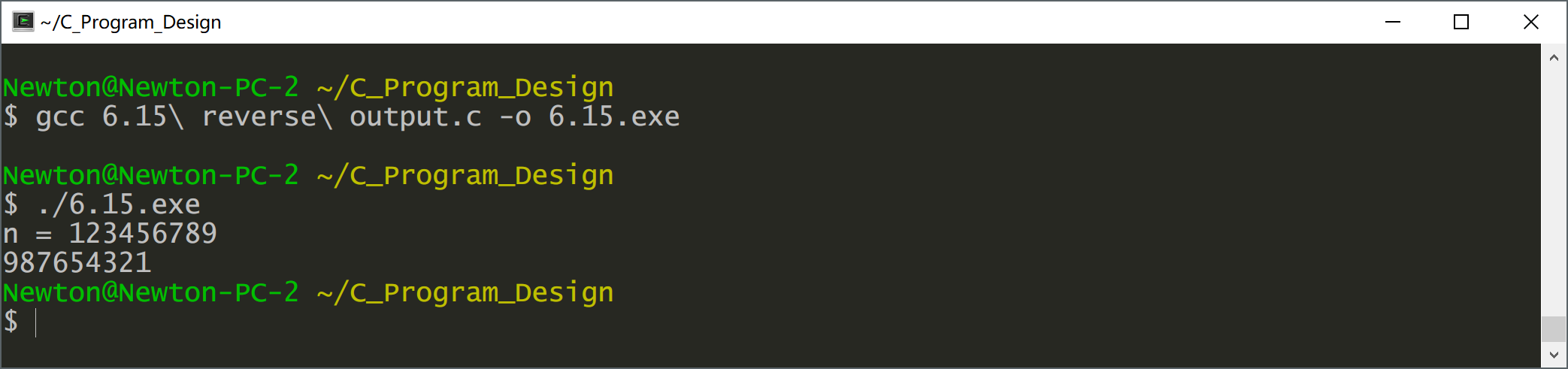
## 反向输出

写一递归函数，将读入的整数按位分开后以相反顺序输出。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | /\*  \* filename: 6.15 reverse output.c  \* property: exercise  \*/  #include <stdio.h>  int f**(**int n**)** **{**  int z**;**  **if** **(**n **<** 10**)** **{**  printf**(**"%d"**,** n**);**  **return** 0**;**  **}**  **else** **{**  z **=** n **%** 10**;**  printf**(**"%d"**,** z**);**  n**=(**n **-** n **%** 10**)** **/** 10**;**  **return** f**(**n**);**  **}**  **}**  int main**()** **{**  int n**;**  printf**(**"n = "**);**  scanf**(**"%d"**,** **&**n**);**  f**(**n**);**  **return** 0**;**  **}** |

### 运行结果

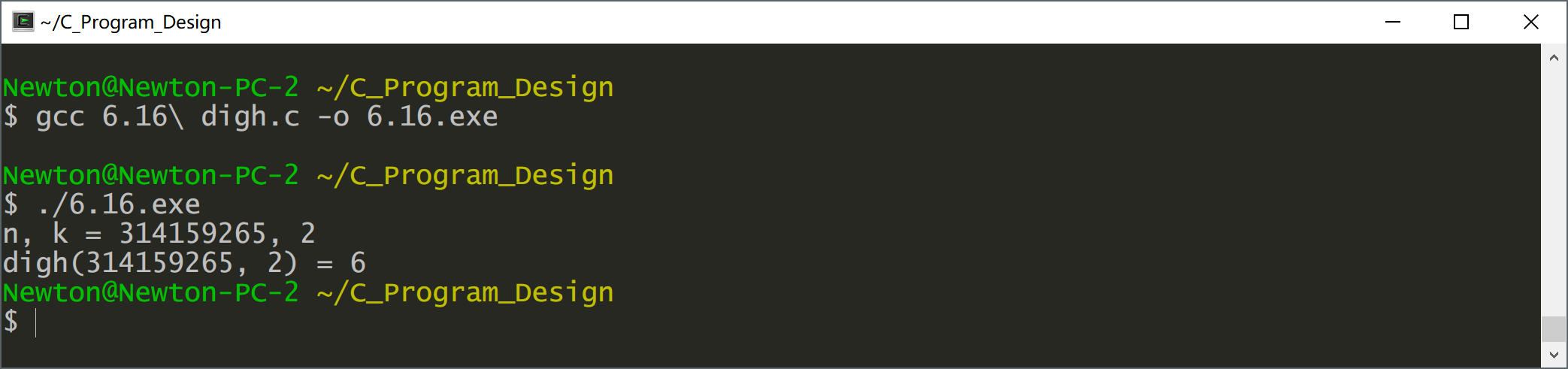


写一函数，它将回送整数从右边开始的第个数字的值，例如，。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | /\*  \* filename: 6.16 digh.c  \* property: exercise  \*/  #include <stdio.h>  int digh**(**int n**,** int k**)** **{**  int y**;**  int t**;**  **for** **(**t **=** 1**;** t **<=** k**;** t**++)** **{**  y **=** n **%** 10**;**  n **=** **(**n **-** y**)** **/** 10**;**  **}**  **return** y**;**  **}**  int main**()** **{**  int n**,** k**;**  printf**(**"n, k = "**);**  scanf**(**"%d, %d"**,** **&**n**,** **&**k**);**  printf**(**"digh(%d, %d) = %d"**,** n**,** k**,** digh**(**n**,** k**));**  **return** 0**;**  **}** |

### 运行结果



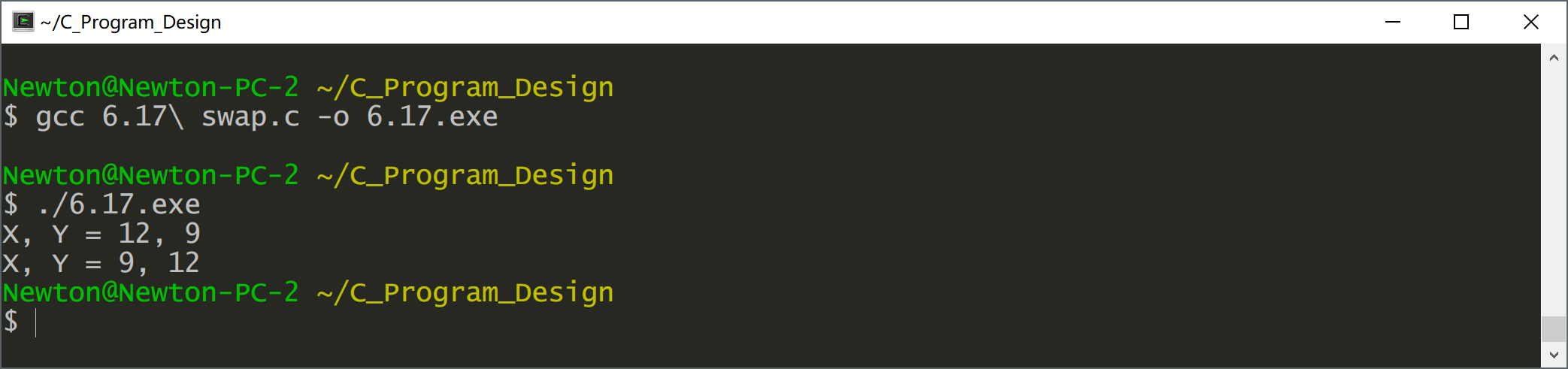
## 两数对换

定义一个宏，完成对两个整数的交换。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | /\*  \* filename: 6.17 swap.c  \* property: exercise  \*/  #include <stdio.h>  int X**,** Y**;**  void swap**(**int x**,** int y**)** **{**  Y **=** x**;**  X **=** y**;**  **}**  int main**()** **{**  printf**(**"X, Y = "**);**  scanf**(**"%d, %d"**,** **&**X**,** **&**Y**);**  swap**(**X**,** Y**);**  printf**(**"X, Y = %d, %d"**,** X**,** Y**);**  **return** 0**;**  **}** |

### 运行结果



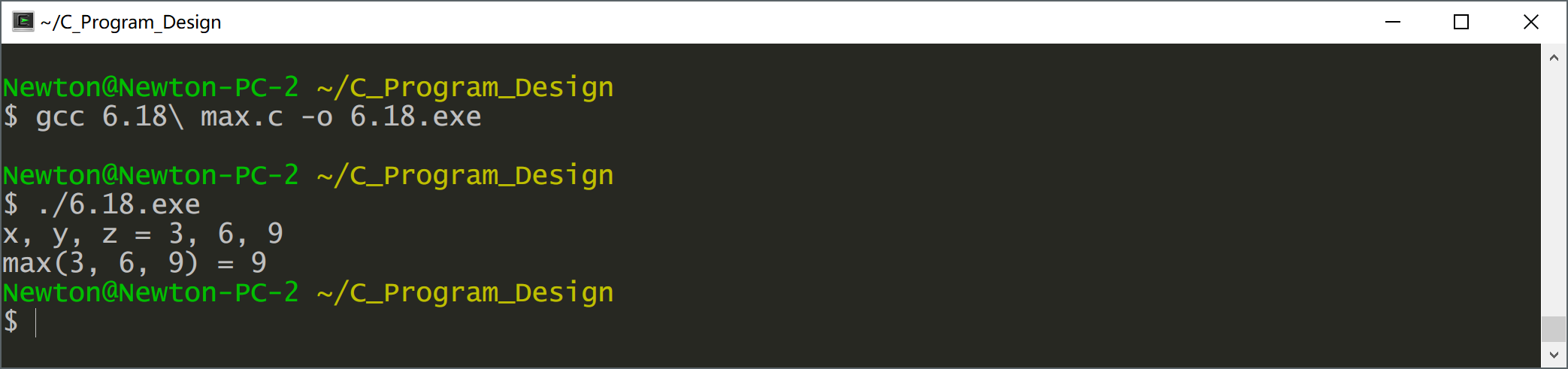
## 三数最大值宏

定义一个宏从三个数中找出最大数。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | /\*  \* filename: 6.18 max.c  \* property: exercise  \*/  #include <stdio.h>  #define MAX(a, b, c) ((a > b? a : b) > c? (a > b? a : b) : c)  int main**()** **{**  int x**,** y**,** z**;**  printf**(**"x, y, z = "**);**  scanf**(**"%d, %d, %d"**,** **&**x**,** **&**y**,** **&**z**);**  printf**(**"max(%d, %d, %d) = %d"**,** x**,** y**,** z**,** MAX**(**x**,** y**,** z**));**  **return** 0**;**  **}** |

### 运行结果



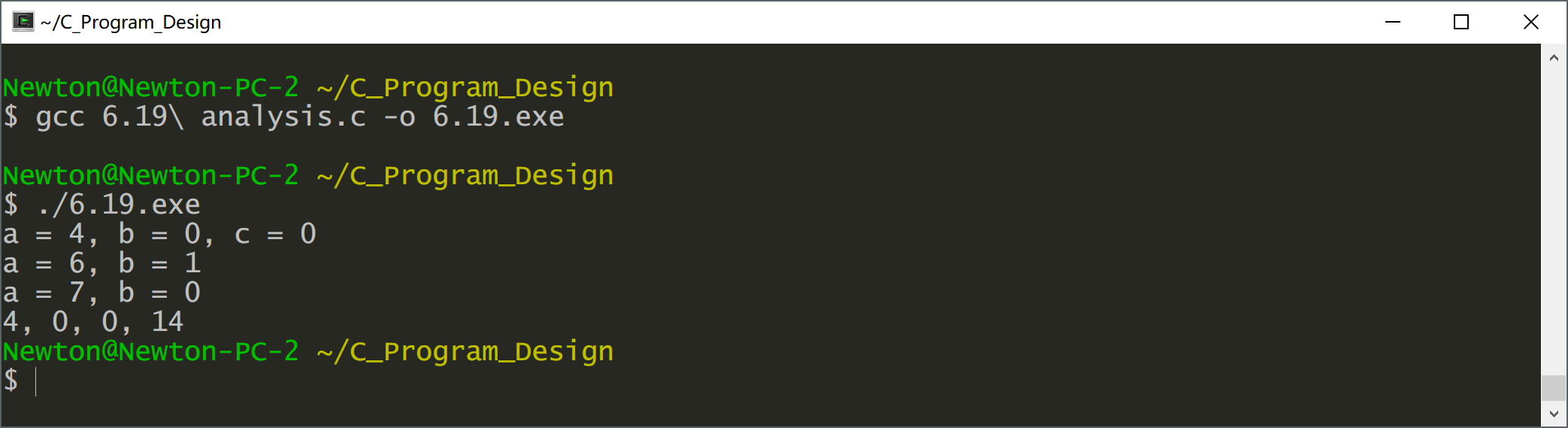
## 代码分析

先静态阅读以下程序，然后上机运行程序，查看运行结果是否与你阅读的结果一致？不一致的原因何在？

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | /\*  \* filename: 6.19 analysis.c  \* property: analysis  \*/  #include <stdio.h>  int a **=** 3**,** b **=** 5**,** c **=** 2**;**  int f**(**int a**,** int b**)** **{**  printf**(**"a = %d, b = %d\n"**,** a**,** b**);**  a**++;**  b**--;**  printf**(**"a = %d, b = %d\n"**,** a**,** b**);**  c **=** a **+** b**;**  **return** a **+** b **+** c**;**  **}**  int main**()** **{**  int a **=** 4**,** b**,** c**,** k**;**  printf**(**"a = %d, b = %d, c = %d\n"**,** a**,** b**,** c**);** // added command  k **=** f**(**a **+** 2**,** b **+** 1**);**  printf**(**"%d, %d, %d, %d"**,** a**,** b**,** c**,** k**);**  **return** 0**;**  **}** |

### 运行结果



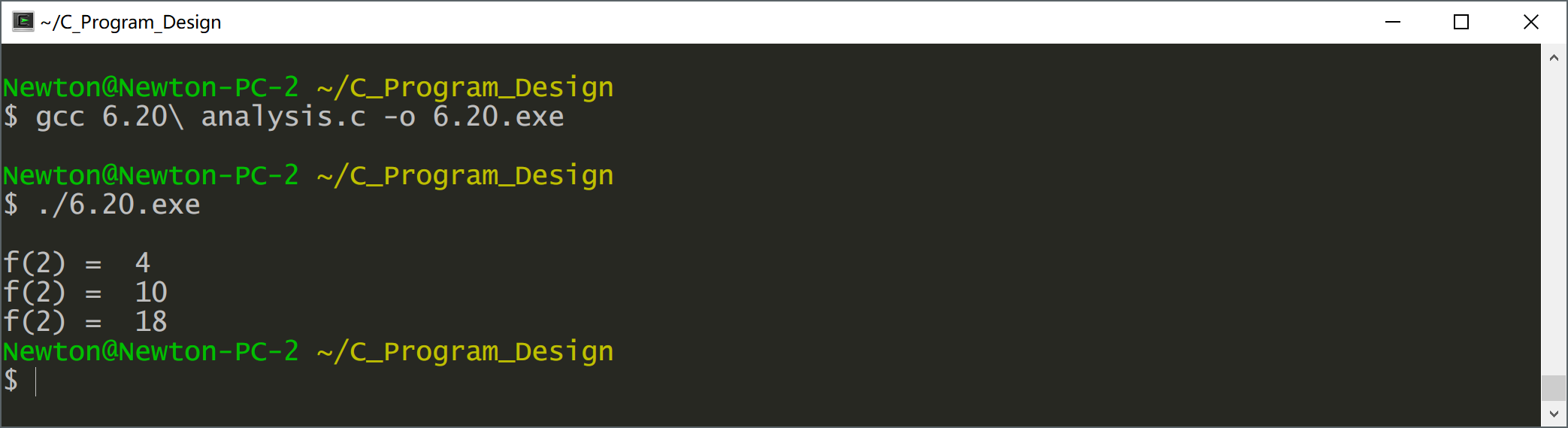
通过添加几个输出语句，可以大致推断C的语言属性：凡是全局变量在某个函数中被重新定义，那就覆盖全局变量。值传递本质上还是一种赋值，被赋值对象也是函数中被定义的变量，如果和全局函数同名必然会将全局函数再度覆盖。如果不覆盖，那就引用全局变量。这个是C的namespace管理机制。

静态分析以下程序的执行结果，然后上机运行程序，将分析结果与运行结果加以对比，从中领会静态局部变量的含义及用法。

### 程序代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | /\*  \* filename: 6.20 analysis.c  \* property； analysis  \*/  #include <stdio.h>  int f**(**int x**)** **{**  static int f **=** 0**,** y **=** 0**;**  **if** **(**f **==** 0**)**  y **+=** 2 **\*** x**;**  **else**  **if(**f **==** 1**)**  y **+=** 3 **\*** x**;**  **else**  y **+=** 4 **\*** x**;**  f**++;**  **return** y**;**  **}**  int main**()** **{**  printf**(**"\nf(2) = %d"**,** f**(**2**));**  printf**(**"\nf(2) = %d"**,** f**(**2**));**  printf**(**"\nf(2) = %d"**,** f**(**2**));**  **return** 0**;**  **}** |

### 运行结果



static修饰符可用来修饰函数或者变量。

1. 全局静态变量：在全局变量之前加上关键字static，全局变量就被定义成为一个全局静态变量。

（1）内存中的位置：静态存储区（静态存储区在整个程序运行期间都存在）

（2）初始化：未经初始化的全局静态变量会被程序自动初始化为0（自动对象的值是任意的，除非他被显示初始化）

（3）作用域：全局静态变量在声明他的文件之外是不可见的。准确地讲从定义之处开始到文件结尾。

定义全局静态变量的好处：

（1）不会被其他文件所访问，修改

（2）其他文件中可以使用相同名字的变量，不会发生冲突。

2. 局部静态变量：在局部变量之前加上关键字static，局部变量就被定义成为一个局部静态变量。

（1）内存中的位置：静态存储区

（2）初始化：未经初始化的局部静态变量会被程序自动初始化为0（自动对象的值是任意的，除非他被显示初始化）

（3）作用域：作用域仍为局部作用域，当定义它的函数或者语句块结束的时候，作用域随之结束。

当static用来修饰局部变量的时候，它就改变了局部变量的存储位置，从原来的栈中存放改为静态存储区。**但是局部静态变量在离开作用域之后，并没有被销毁，而是仍然驻留在内存当中，直到程序结束，只不过我们不能再对他进行访问。**当static用来修饰全局变量的时候，它就改变了全局变量的作用域（在声明他的文件之外是不可见的），但是没有改变它的存放位置，还是在静态存储区中。

# 实验总结

当我再次完成这次报告的时候，已经是写过数个体量较大的工程了。不过在做大工程的时候，往往还是把视野局限在完成功能上，对于一些C的修饰语句从来都没有使用过，对C的内存分配仅是一知半解。经过对本次实验中的几个analysis属性的代码进行静态分析，我回忆起了往昔一些学过但是不曾用过的东西，这在我做过的很多项目里都是可以起优化作用的。

模块化涉及函数模块化与功能模块化，很多复杂的工程都是把复杂的系统拆分成一个一个的功能模块，源代码放置在不同的文件夹里。这一点在本实验中没有体现出来，我决定放到后期的综合训练中，用GNU make进行编译管理，具体代码内容的安排，也参见后期的一些工程。

经过对《UNIX环境高级编程》[1]这本书的学习，还有诸如*Harley Hahn's Guide to Unix and Linux*[2]这本书的阅读，我觉得基于Shell的UNIX环境似乎是最适合新手学习的。

本次实验，集中主要精力，在以前版本的基础上，对文档结构进行了重整，看起来自然了很多，目录也规范了很多。有关编程的规范性问题，参考林锐高质量C/C++编程指南的第一版[3]。

# 参考文献

1. Stevens, W.R. and S.A. Rago, *UNIX环境高级编程*. 2nd ed. 2005, 北京: 人民邮电出版社.

2. Hahn, H., *Harley Hahn's Guide to Unix and Linux*. 2009, New York: McGraw-Hill.

3. **林锐**, *高质量 C++/C 编程指南*. 1.0 ed. 2001.

# 教师评语