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：指针程序设计(1) |  |  |
| 实验编号：No.07 | 实验日期：2017年8月20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

1. 了掌握变量的指针及其基本用法。
2. 掌握一维数组的指针及其基本用法。
3. 掌握指针变量作为函数的参数时，参数的传递过程及其用法。

# 实验环境

Windows10 Pro Workstation 17134.228；

Cygwin GCC编译器。

# 实验内容

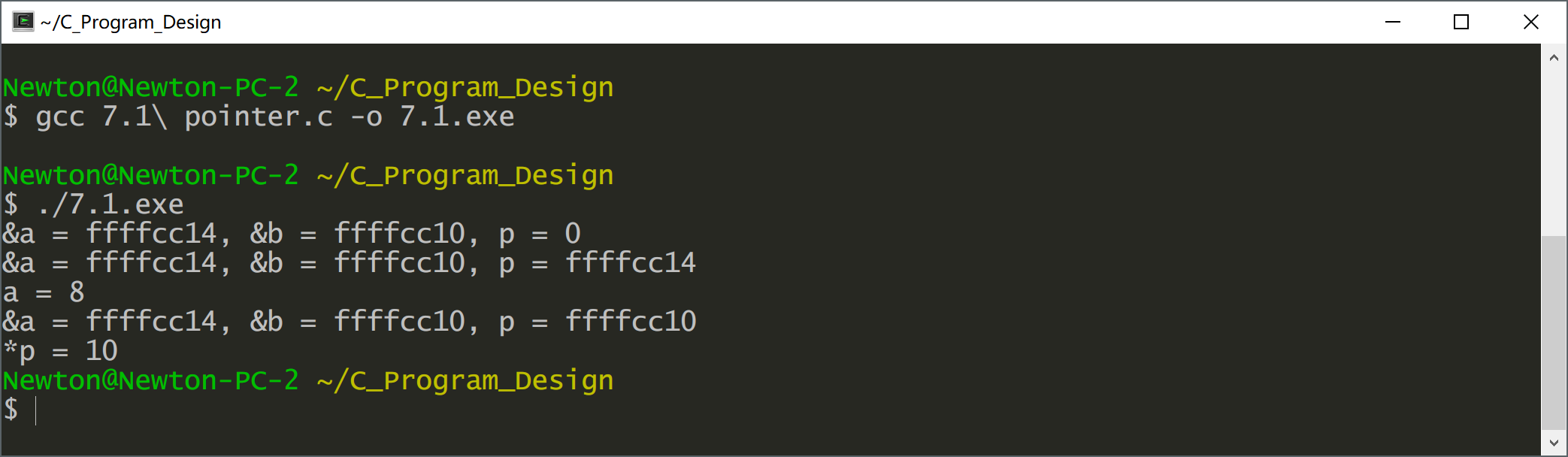
## 指针值观测

对以下程序进行单步运行,并从中了解变量的指针和指针变量的概念。

### 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: 7.1 pointer.c  \* property: test  \*/  #include <stdio.h>  int main**()** **{**  int a **=** 5**,** b **=** 5**,** **\***p**;**  printf**(**"&a = %x, &b = %x, p = %x\n"**,** **&**a**,** **&**b**,** p**);**  p **=** **&**a**;**  **\***p **=** 8**;**  printf**(**"&a = %x, &b = %x, p = %x\n"**,** **&**a**,** **&**b**,** p**);**  printf**(**"a = %d\n"**,** a**);**  p **=** **&**b**;**  b **=** 10**;**  printf**(**"&a = %x, &b = %x, p = %x\n"**,** **&**a**,** **&**b**,** p**);**  printf**(**"\*p = %d"**,** **\***p**);**  **return** 0**;**  **}** |

### 运行结果



按以下步骤操作：

1. 输入程序后，连续按三次F8，使绿条停留在P=&a语句行上。

②用<Ctrl F7>操作分别将a，b，&a，&b，p及\*p显示出来。

③查看观察窗口中的内容，可发现此时a、b己有确定的地址(&a和&b)和确定的值，而p还没有确定的值(此时语句p=&a还未执行)，即p还没有明确的指向，因而它所指向的内存单元(\*p)中的内容也是不确定的。

④按F8往下执行一步后再查看观察窗中的内容，可发现p已有确定的值，它与&a的值一致，说明p中存放了变量a的地址，也就是说p是指向变量a的指针变量。同时可发现，\*p的内容与a的内容一致，即p所指向的内存单元中的内容就是a的内容；从而可以理解\*p等效于a，表示同一内存单元。

⑤按F8往下执行一步后再查看观察窗中的内容，可发现\*p和a的内容都已发生变化，从而可理解通过改变指针变量p所指向的内存单元中的内容可以间接地改变a中的内容。

⑥再按F8往下执行一步，可发现p的值己发生变化，它与&b的值一致，说明p已经是指向变量b的指针变量，它不再是指向a，\*p的内容也已变为b的内容，从而可理解指针变量的指向是随时可以改变的。

⑦再按F8往下执行一步，可发现，b的值和\*p的值都已发生变化，即改变b的内容就等于改变指针变量p所指向的内存单元中的内容。

## 指针值与指向值观测

单步运行以下程序，观察&a[0]，&a[i]和P的变化，然后回答以下问题:

①程序的功能是什么？

答：求数组元素的和。

②在开始进入循环体之前，p指向谁？

答：指向数组的首地址。

③循环每增加一次，p的值(地址)增加多少？它指向谁?

答：增加sizeof(int)；指向数组的下一个元素。

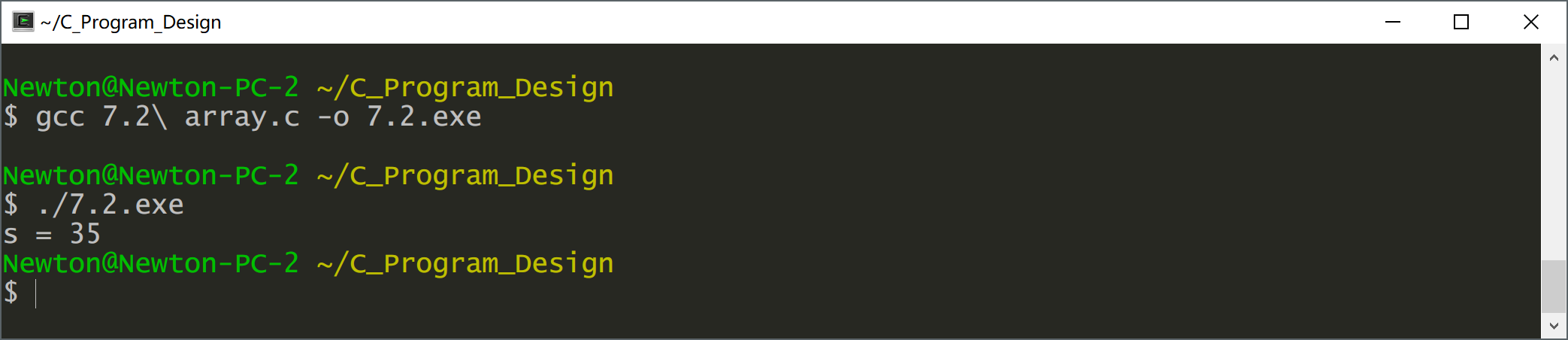
④退出循环后，p指向谁？

答：指向内存中，数组变量最后一个元素的后一个变量。

⑤你是否初步掌握了通过指针引用数组元素的方法？

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | /\*  \* filename: 7.2 array.c  \* property: test  \*/  #include <stdio.h>  int main**()** **{**  int i**,** **\***p**,** s **=** 0**;**  int a**[**5**]** **=** **{**5**,** 6**,** 7**,** 8**,** 9**};**  p **=** a**;**  **for(**i **=** 0**;** i **<** 5**;** i**++,** p**++)** **{**  s **+=** **\***p**;**  **}**  printf**(**"s = %d"**,** s**);**  **return** 0**;**  **}** |

### 运行结果



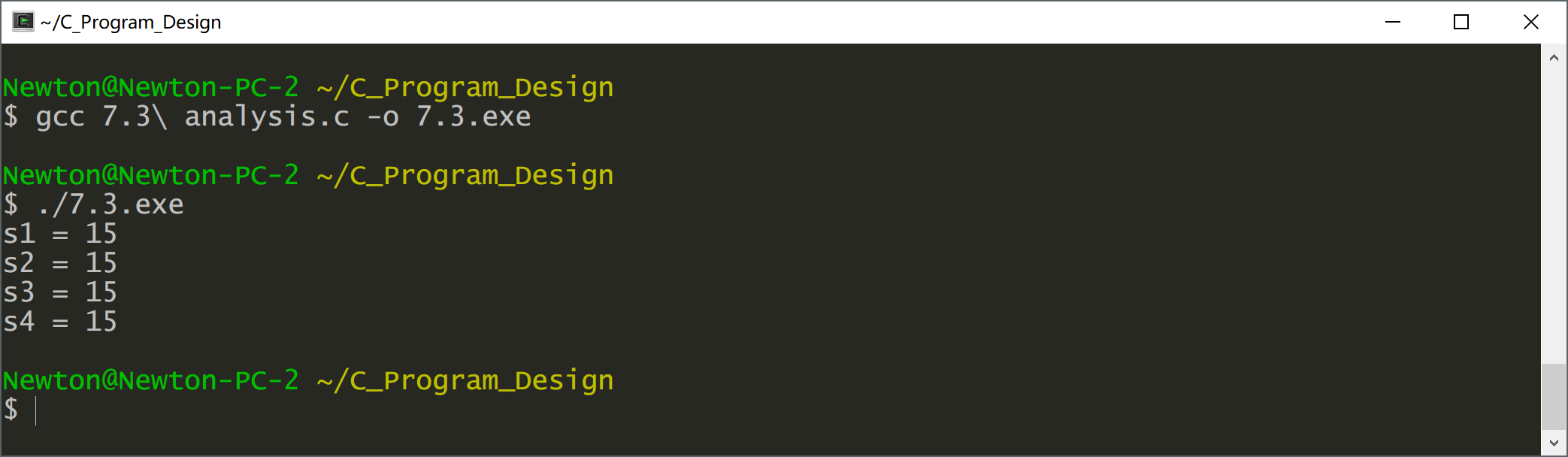
## 指针引用练习

先分析以下程序的运行结果，然后上机验证，并通过此例掌握通过指针变量引用数组元素的各种方法。

### 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  23  24  25  26  27 | /\*  \* filename: 7.3 analysis.c  \* property: analysis  \*/  #include <stdio.h>  int main**()** **{**  int i**,**s1 **=** 0**,** s2 **=** 0**,** s3 **=** 0**,** s4 **=** 0**,** **\***p**;**  int a**[**5**]** **=** **{**1**,** 2**,** 3**,** 4**,** 5**};**  p **=** a**;**  **for** **(**i **=** 0**;** i **<** 5**;** i**++)** **{**  s1 **+=** p**[**i**];**  **}**  **for** **(**i **=** 0**;** i **<** 5**;** i**++)** **{**  s2 **+=** **\*(**p **+** i**);**  **}**  **for** **(**p **=** a**;** p **<** a **+** 5**;** p**++)** **{**  s3 **+=** **\***p**;**  **}**  p **=** a**;**  **for** **(**i **=** 0**;** i **<** 5**;** i**++)** **{**  s4 **+=** **\***p**++;**  **}**  printf**(**"s1 = %d\ns2 = %d\ns3 = %d\ns4 = %d\n"**,** s1**,** s2**,** s3**,** s4**);**  **return** 0**;**  **}** |

### 运行结果



## 一元二次方程求根

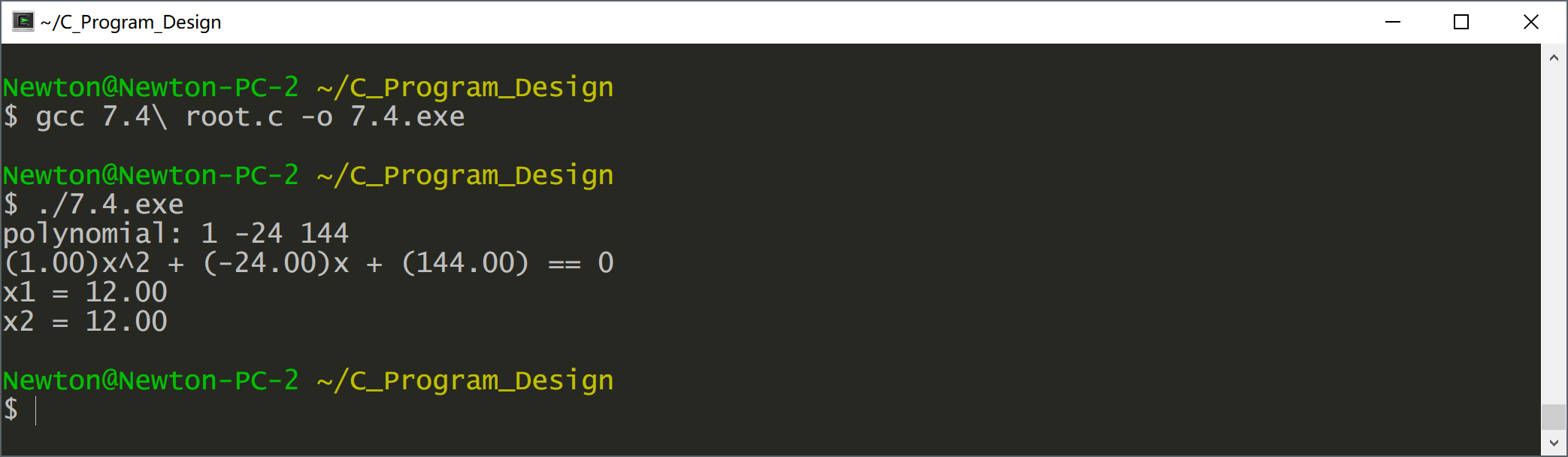
编写函数实现计算一元二次方程的两个实根，然后编写主函数调用此函数。

要求：在同一个函数内求出方程的两个实根，此函数不准使用全局变量进行数据传递，也不能使用return语句，只能通过指针进行数据传递。

### 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 | /\*  \* filename: 7.4 root.c  \* property: exercise  \*/  #include <stdio.h>  #include <math.h>  void root**(**float **\***a**,** float **\***b**,** float **\***c**,** float **\***x1**,** float **\***x2**)** **{**  float d **=** **(\***b**)** **\*** **(\***b**)** **-** 4 **\*** **(\***a**)** **\*** **(\***c**);**  **if** **(**d **<** 0**)** **{**  printf**(**"no real root;"**);**  **}**  **else** **{**  **\***x1 **=** **(-(\***b**)** **+** sqrt**(**d**))** **/** **(**2 **\*** **(\***a**));**  **\***x2 **=** **(-(\***b**)** **-** sqrt**(**d**))** **/** **(**2 **\*** **(\***a**));**  **}**  **}**  int main**()** **{**  printf**(**"polynomial: "**);**  float a**,** b**,** c**;**  float x1**,** x2**;**  scanf**(**"%f %f %f"**,** **&**a**,** **&**b**,** **&**c**);**  printf**(**"(%3.2f)x^2 + (%3.2f)x + (%3.2f) == 0\n"**,** a**,** b**,** c**);**  root**(&**a**,** **&**b**,** **&**c**,** **&**x1**,** **&**x2**);**  printf**(**"x1 = %3.2f\nx2 = %3.2f\n"**,** x1**,** x2**);**    **return** 0**;**  **}** |

### 运行结果



## 逆序输出序列

编写函数，将个数按原来的顺序的逆序排列（要求用指针实现），然后编写主函数完成：

①输入10个数；

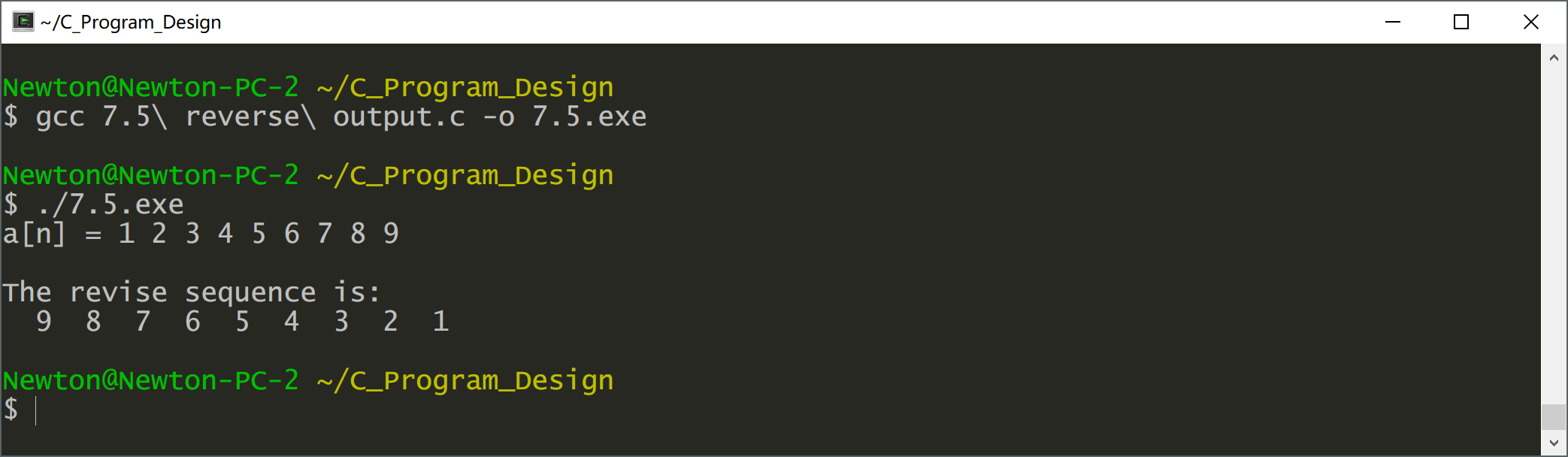
②调用此函数进行重排；

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 | /\*  \* filename: 7.5 reverse output.c  \* property: exercise  \*/  #include <stdio.h>  #define NUM 256  void reverse**(**int **\***a**,** int count**)** **{**  int i**;**  int tmp**;**  **for** **(**i **=** 0**;** i **<=** count **/** 2**;** i**++)** **{**  tmp **=** a**[**i**];**  a**[**i**]** **=** a**[**count **-** i**];**  a**[**count **-** i**]** **=** tmp**;**  **}**  **}**  int main**()** **{**  int i**,** a**[**NUM**];**  int **\***p**,** count **=** 0**;**  p **=** a**;**  char c**;**  printf**(**"a[n] = "**);**  **for(**i **=** 0**;** i **<** NUM**;** i**++)** **{**  scanf**(**"%d%c,"**,** **&**a**[**i**],** **&**c**);**  **if** **(**c **!=** ' '**)** **{**  **break;**  **}**  **else** **{**  count **+=** 1**;**  **}**  **}**  reverse**(**a**,** count**);**  printf**(**"The revise sequence is:\n"**);**  **for(**i **=** 0**;** i **<=** count**;** i**++)** **{**  printf**(**"%3d"**,** **\*(**p **+** i**));**  **}**  printf**(**"\n"**);**  **return** 0**;**  **}** |

### 运行结果



# 实验总结

指针是高级数据结构的基石。通过指针，可以更加方便地对内存中的数据进行操作，写出比较好的分离式模块。通过迭代器模式，可以将算法与数据结构分开，而这也是通过指针来实现的。

随着时间的推移，我的编译器选择历经轮转，从最初的古老的TC2.0，到Code::Blocks集成开发环境，再到Visual Studio 2017，最终还是回到了GNU平台上来，使用开源的一套库进行实验。在此期间，云南大学也从一个普通的211大学跻身双一流大学行列，高级语言程序设计这门课程是否也该升级一下？TC2.0这个编译器，界面十分古朴，在几十年前绝对算是一流的软件，但是现在，确实落后了，这主要是因为它无法编译在64位系统下运行的程序。但是TC2.0有很多的优势，比如完全可视化的编译过程，不会生成很多附加文件，这一点Visual Studio就太过专业化。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