



**程序设计实验报告**

学 院

专 业

年级班别

学 号

学生姓名

指导教师

2018年 10 月

# 实验1

## 实验目的

1. 理解编程语言的特点；
2. 初步掌握在计算机上运行程序的方法；
3. 理解算法的概念和特性；
4. 能够用N-S流程图表示算法。
5. 在掌握基本数据类型的表示方法、运算符和表达式概念的基础上，能够进行简单的程序编写。

## 实验工具

Tubro C 2.0/3.0

## 编写要求

对每个实验题目，需要提供如下内容：

1. 程序流程图（应采用NS图或传统流程图）；
2. 具体程序代码（具有良好的程序设计风格，如程序模块化、加注释、缩进的书写格式）；
3. 实验运行结果（对比测试用例）；
4. 对运行情况所做的分析以及调试程序所取得的经验收获。

## 排版要求

1. 标题四号宋体，正文小四宋体，段间距1.5倍；
2. 除封面以外，其他内容双面打印。

## 实验题目

1. 从键盘输入四个浮点数w,x,y,z，把4个数由大到小输出，文件名 A1.c

测试用例： （1）3, 2, 1.0 ,4.2（2）3.1, -1,-3.14,0 （3）0,9.2,-1 0

**NS图 ：**

|  |  |  |  |
| --- | --- | --- | --- |
| int i, j, k；float temp, a[4] | | | |
| for(i = 0; i < 4; i++) | | | |
|  | 输入a[i] | | |
| for (i = 0; i < 3; i++) | | | |
|  | k=i | | |
| for (j = i + 1; j < 4; j++)  a[k]<a[j]&&fabs(a[k]-a[j])>1e-6 | | |
|  | Y | N |
| k=j |  |
|  | N  Y  k!=i | |
| temp=a[i]  a[i]=a[k]  a[k]=temp | |  |
| 输出数组a | | | |

**A1.c**

#include <stdio.h>

#include <math.h>

int main()

{

int i, j, k;

float a[4], temp;

for (i = 0; i < 4; i++)

scanf("%f", &a[i]);

for (i = 0; i < 3; i++)

{

k = i;

for (j = i + 1; j < 4; j++)

if (a[k] < a[j] && fabs(a[k] - a[j]) > 1e-6)

k = j;

if (k != i)

{

temp = a[i];

a[i] = a[k];

a[k] = temp;

}

}

for (i = 0; i < 4; i++)

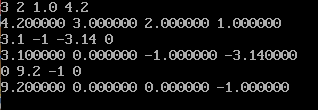
printf("%f ", a[i]);

printf("\n");

return 0;

}

**A1.c运行结果**



1. 从键盘输入三个整数x,y,z，请把3个数由大到小输出,文件名 A2.c

测试用例： （1）1 2 3（2）2 -1 -3 （3）0 -1 2

**NS图**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| int i, j, k, temp, a[3] | | | | |
| for(i = 0; i < 3; i++) | | | | |
|  | | 输入a[i] | | |
| for (i = 0; i < 2; i++) | | | | |
|  | k=i | | | |
| for (j = i + 1; j < 3; j++) | | | |
|  | | Y  a[k]<a[j] | N |
| k=j |  |
|  | | N  Y  k!=i | |
| temp=a[i]  a[i]=a[k]  a[k]=temp | | |  |
| 输出数组a | | | | |

**A2.c**

#include <stdio.h>

int main()

{

int i, j, k, temp;

int a[3];

printf("Please enter three integer numbers: ");

for (i = 0; i < 3; i++)

scanf("%d", &a[i]);

for (i = 0; i < 2; i++)

{

k = i;

for (j = i + 1; j < 3; j++)

if (a[k] < a[j])

k = j;

if (k != i)

{

temp = a[i];

a[i] = a[k];

a[k] = temp;

}

}

for (i = 0; i < 3; i++)

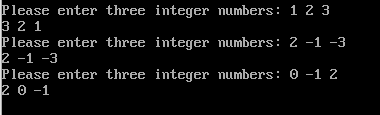
printf("%d ", a[i]);

printf("\n");

return 0;

}

**A2.c运行结果**



1. 从键盘上输入‘0’至‘9’中任何一个字母，要求将其转化成整数输出。文件名 A3.c, 不允许直接将字符素数据直接输出

测试用例: （1）输入字符‘9’，输出数字9

（2）输入字符‘3’，输出数字3

**NS图**

|  |
| --- |
| 输入一个‘0’到‘9’的字符ch |
| ch=ch-‘0’ |
| 输出ch |

**A3.c**

#include <stdio.h>

int main()

{

printf("%d\n", getchar() - '0');

return 0;

}

**A3.c运行结果**



**总结：**第一题由于是浮点数比较，所以不能直接用不等号来比较，还要加一个两数相减的绝对值与一个很小的数之间的比较。第三题由于只输出一个数，所以可以输入输出用一个语句就能搞定，免去了再设一个变量。

# 实验2

## 实验目的

1. 理解基本的语句，如控制语句、赋值语句等；
2. 能够结合输入输出语句设计编写简单的程序；
3. 在了解关系运算符和关系表达式，逻辑运算符和逻辑表达式的基础上，能够编写具有选择结构的程序；
4. 能够区别多种循环，具备实现循环的能力，编写具有循环嵌套的程序。

## 实验工具

Tubro C 2.0/3.0

## 编写要求

对每个实验题目，需要提供如下内容：

1. 程序流程图（应采用NS图或传统流程图）；
2. 具体程序代码（具有良好的程序设计风格，如程序模块化、加注释、缩进的书写格式）；
3. 实验运行结果（对比测试用例）；
4. 对运行情况所做的分析以及调试程序所取得的经验收获。

## 排版要求

1. 标题四号宋体，正文小四宋体，段间距1.5倍；
2. 除封面以外，其他内容双面打印。

## 实验题目

1． 接收一个六位正整数。编写一个程序，将该数的每一位数字相加并显示结果，最后将这六位数从大到小排序输出，比如314500，输出和13，排序后成为一个新的数543100。若输入的不是六位数，则要提示出错。用循环实现。

测试用例：(1)123456 (2)890100 (3)12

**NS图：**

**fun函数**

long int fun(int \*a, int n, int x)

|  |  |
| --- | --- |
| Y  n <= 1  N | |
| return a[0] + fun(a + 1 , n - 1, x) \* x; | return a[0]; |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| long int num; | | | | |
| Int a[6],sum=0,i=0,j,k,temp; | | | | |
| 输入num | | | | |
| num >= 100000 && num < 1000000  N  Y | | | | |
| while(num) | | | | 输入错误 |
|  | a[i] = num % 10; | | |
| num /= 10; | | |
| sum += a[i++]; | | |
|  | for (i = 0; i < 5; i++) | | |
|  | k=i | | |
| for (j = i + 1; j < 6; j++) | | |
|  | Y  a[k]<a[j] | N |
| k=j |  |
|  | N  Y  k!=i | |
| temp=a[i]  a[i]=a[k]  a[k]=temp | |  |
| num=fun(a,6,10); | | | |
| 输出sum，num。 | | | | |

**main函数**

**代码：**

#include <stdio.h>

long int fun(int \*a, int n, int x)//用递归使num中的数从大到小排序

{

if (n <= 1)

return a[0];

else

return a[0] + ( fun (a + 1 , n - 1, x)) \* x;

}

int main()

{

long int num;

int a[6], sum = 0, temp, i = 0, j, k;

printf("Please enter an integer number: ");

scanf("%ld", &num);

if (num >= 100000 && num < 1000000)

{

while (num)//将位数分离

{

a[i] = num % 10;

num /= 10;

sum += a[i++];

}

for (i = 0; i < 5; i++)

{//将位数排序

k = i;

for (j = i+1; j < 6; j++)

if (a[k] > a[j])

k = j;

if (k != i)

{

temp = a[i];

a[i] = a[k];

a[k] = temp;

}

}

num=fun(a, 6, 10);

printf("%d\n", sum);

printf("%ld\n", num);

}

else

{

fprintf(stderr, "Improper value of num\n");

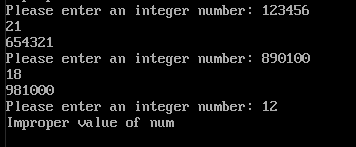
exit(1);

}

return 0;

}

**运行结果：**

**总结：**

这道题如果要从屏幕直接输出一道位数从大到小的数是可以直接输出一个数组的，但是那样只是在屏幕看上去是一个数，其实是多个元素拼接起来的。所以为了让num按题目那样输出，我写了一个递归函数，通过递归实现了num的转变。

# 实验3

## 实验目的

1. 能够定义和引用一维数组、二维数组和字符数组；
2. 能够利用数组来设计相关程序；
3. 掌握函数的定义和调用方法、变量的作用域和存储方式等概念；
4. 理解函数调用时参数传递的过程；
5. 能够区分值传递方式和地址传递；
6. 能够用函数实现模块化的程序设计。

## 实验工具

Tubro C 2.0/3.0

## 编写要求

对每个实验题目，需要提供如下内容：

1. 程序流程图（应采用NS图或传统流程图）；
2. 具体程序代码（具有良好的程序设计风格，如程序模块化、加注释、缩进的书写格式）；
3. 实验运行结果（对比测试用例）；
4. 对运行情况所做的分析以及调试程序所取得的经验收获。

## 排版要求

1. 标题四号宋体，正文小四宋体，段间距1.5倍；
2. 除封面以外，其他内容双面打印。

## 实验题目

1． 有两个整型数组，一个长度为6，一个长度为5,将两个数组合并,存放在a和b数组中，合并和存放规则如下

(1)先采用冒泡法将a数组按照升序进行排序，输出a数组的所有元素

(2)先采用冒泡法将b数组按照升序进行排序，输出b数组的所有元素

(3)将b数组的元素插入到a数组当中，若是最小放在最前面，否则插入到一个合适的位置

(4)若a数组不够存储，则存放到b数组中，剩下的数也是升序排序。

(5)不允许借组第三个数组来实现功能

(6)将合并后a数组和b数组分别输出,文件名取名为P1.c

比如：a {8,3,1,5,4,9} b {6,3,2,-1,10}

(1)排序 a {1,3,4,5,8,9} b{-1,2,3,6,10}

(2)插入合并a {-1,1,2,3,3,4} b{5,6,8,9,10}

测试用例：(1) a:{8,3,1,5,4,9} b:{6,3,2,-1,10}

合并后结果 a:{-1,1,2,3,3,4} b:{5,6,8,9,10}

(2) a:{11,8,7,6,4,3} b:{2,1,5,3,9}

合并后结果 a:{1,2,3,3,4,5} b:{6,7,8,9,11}

(3) a:{0,-3,8,100,20,30,}，b:{4,10,101,-5,0}

合并后结果 a:{-5,-3,0,0,4,8,}，b:{10,20,30,100,101}

**NS图**

**void sort(int \*a, int n);**

|  |  |  |  |
| --- | --- | --- | --- |
| int i, j, temp | | | |
| for（i= 0; i < n - 1; i++） | | | |
|  | for (j = 0; j < n – 1 - i; j++) | | |
|  | N  Y  a[j]>a[j+1] | |
| temp=a[i]  a[i]=a[j]  a[j]=temp |  |

**main函数**

|  |  |  |  |
| --- | --- | --- | --- |
| int i, j, k, temp, a[6], b[5] | | | |
| 输入数组a, b | | | |
| sort(a, 6） | | | |
| sort(b, 5) | | | |
| 输出数组a,b | | | |
| for (i = 0, i < 5; i++) | | | |
|  | for (j = 0; j < 6; j++) | | |
|  | N  Y  a[j]>b[i] | |
| temp=a[j]  a[j]=b[i]  b[i]=temp |  |
| sort(b, 5) | | | |
| 输出数组a,b | | | |

**代码：**

#include <stdio.h>

void sort(int \*, int);//需多次用到排序，另写一个函数会比较方便

int main()

{

int a[6], b[5], i, j, temp;

for (i = 0; i < 6; i++)

scanf("%d", &a[i]);

for (i = 0; i < 5; i++)

scanf("%d", &b[i]);

sort(a, 6);//对a,b进行冒泡排序

sort(b, 5);

for (i = 0; i < 6; i++)

printf("%d ", a[i]);

printf("\n");

for (i = 0; i < 5; i++)

printf("%d ", b[i]);

printf("\n");

for (i = 0; i < 5; i++)

{

for (j = 0; j < 6; j++)//遍历a，使得a中最大的元素交换到b中

{

if (a[j] > b[i])

{

temp = a[j];

a[j] = b[i];

b[i] = temp;

}

}

}

sort(b, 5);//重新对b进行冒泡排序

for (i = 0; i < 6; i++)

printf("%d ", a[i]);

printf("\n");

for (i = 0; i < 5; i++)

printf("%d ", b[i]);

printf("\n\n");

return 0;

}

void sort(int \*a, int n)

{

int i, j, temp;

for (i = 0; i < n - 1; i++)

{

for (j = 0; j < n - 1 - i; j++)

{

if (a[j] > a[j + 1])

{

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

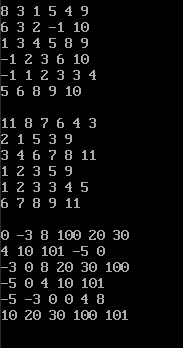
}

}

}

}

**运行结果：**



**总结：**

感觉这次实验也有些不足的地方，可以把合并操作也是要一个函数进行整合。

# 实验4

## 实验目的

1. 学会使用多文件多函数的编程；
2. 了解常见的排序算法；
3. 能够理解和使用指针变量；
4. 能够设计通过指针引用数组和字符串以及指向函数的指针的程序。

## 实验工具

Tubro C 2.0/3.0

## 编写要求

对每个实验题目，需要提供如下内容：

1. 程序流程图（应采用NS图或传统流程图）；
2. 具体程序代码（具有良好的程序设计风格，如程序模块化、加注释、缩进的书写格式）；
3. 实验运行结果（对比测试用例）；
4. 对运行情况所做的分析以及调试程序所取得的经验收获。

## 排版要求

1. 标题四号宋体，正文小四宋体，段间距1.5倍；
2. 除封面以外，其他内容双面打印。

## 实验题目

1． 多文件多函数编程

sort.prj -->sort.exe

有两个整型数组，一个长度为8，一个长度为3,将两个数组合并,存放在数组c中，合并和存放规则如下

s1.c:功能：先采用选择法将a数组按照从大到小进行排序，输出a数组的所有元素。函数原型：void sortA(int a[],int asize);

s2.c:功能：再采用冒泡法将b数组按照从小到大进行排序，输出b数组的所有元素。函数原型：void sortB(int b[],int bsize);

s3.c:功能：将a、b数组的元素进行冒泡法合并排序，使得a、b数组的元素是从小到大排序，最后将a，b数组元素依次拷贝到数组c当中。并将数组c的数据输出。函数原型：void sortAnB2C(int \*a,int asize,int \*b,int bsize,int \*c,int csize);

s4.c功能：主调函数

void main()

{

int a[8],b[3],c[11];

/\*输入a数组\*/；/\*输入b数组\*/；

sortA(a,8); sortB(b,3); sortAnB2C(a,8,b,3,c,11);

}

比如：a {8,3,1,5,4,9,-1,10} b {6,3,2}

(1)排序 a {10,9,8,5,4,3,1,-1} b {2,3,6}

(2)合并排序 a {-1,1,2,3,3,4,5,6} b {8,9,10}

(3) C {-1,1,2,3,3,4,5,6,8,9,10}

测试用例：（1） a:{8,3,1,5,4,9,-1,10} b:{6,3,2}

合并后结果 a:{-1,1,2,3,3,4,5,6} b:{8,9,10}

C:{-1,1,2,3,3,4,5,6,8,9,10}

（2） a:{11,8,7,6,4,3,2,1} b:{5,3,2}

合并后结果 a:{1,2,2,3,3,4,5,6,} b:{7,8,11}

**s1.c---NS图**

|  |  |  |  |
| --- | --- | --- | --- |
| int i, j, k, temp; | | | |
|  | for (i = 0; i < asize - 1; i++) | | |
| k=i | | |
| for (j = i + 1; j < asize; j++) | | |
|  | a[k]<a[j]  Y | N |
| k=j |  |
|  | k != i  N  Y | |
| temp=a[i]  a[i]=a[k]  a[k]=temp | |  |
| 输出数组a | | | |

**S2.c—NS图**

|  |  |  |  |
| --- | --- | --- | --- |
| int i, j, temp; | | | |
|  | for (i = 0; i < bsize - 1; i++) | | |
| for (j = i; j < bsize – 1 -i; j++) | | |
|  | b[j]>b[j+1]  Y | N |
| temp=b[j]  b[j]=b[j+1]  b[j+1]=temp |  |
| 输出数组b | | | |

**S3.c—NS图**

|  |  |  |
| --- | --- | --- |
| int i, j, k**;** | | |
| sort(a, asize); | | |
|  | for (i = 0, j = 0, k = 0; k < csize; k++) | |
| N  Y  j == bsize | |
| c[k] = a[i++]; |  |
| continue; |
| N  Y  i == asize | |
| c[k] = b[j++]; |  |
|  | continue; |
| c[k] = (a[i] < b[j]) ? a[i++] : b[j++]; | |
| 输出数组c | | |

**void sort(int a[], int n);**

|  |  |  |  |
| --- | --- | --- | --- |
| int i, j, temp; | | | |
|  | for (i = 0; i < n - 1; i++) | | |
| for (j = 0; j < n – 1 - i; j++) | | |
|  | a[j]>a[j+1]  Y | N |
|  | temp=a[j]  a[j]=a[j+1]  a[j+1]=temp |  |

**s4.c---NS图**

|  |
| --- |
| int a[8], b[3], c[11]; |
| 输入a[8], b[3]的元素 |
| sortA（a, 8） |
| sortB(b, 3) |
| sortAnB2C(a, 8, b, 3, c, 11) |

**代码：**

**s1.c**

void sortA(int a[], int asize)

{

int i, j, k, temp;

for (i = 0; i < asize - 1; i++)

{

k = i;

for (j = i + 1; j < asize; j++)

if (a[k] < a[j])

k = j;

if (k != i)

{

temp = a[k];

a[k] = a[i];

a[i] = temp;

}

}

for (i = 0; i < asize; i++)

printf("%d ", a[i]);

printf("\n");

}

**s2.c**

void sortB(int b[],int bsize)

{

int i, j, temp;

for (i = 0; i < bsize - 1; i++)

for (j = 0; j < bsize – i - 1; j++)

if(b[j] > b[j + 1])

{

temp = b[j];

b[j] = b[j + 1];

b[j + 1] = temp;

}

for (i = 0; i < bsize; i++)

printf("%d ", b[i]);

printf("\n");

}

**s3.c**

void sortAnB2C(int \*a, int asize, int \*b,int bsize,

int \*c, int csize)

{

int i, j, k;

void sort(int [], int);

sort(a, asize);

for (i = 0, j = 0, k = 0; k < csize; k++)

{//使用归并排序使两个数组合并

if (i == asize)

{

c[k] = b[j++];

continue;

}

if (j == bsize)

{

c[k] = a[i++];

continue;

}

c[k] = (a[i] < b[j]) ? a[i++] : b[j++];

}

for (i = 0; i < csize; i++)

printf("%d ", c[i]);

printf("\n");

}

void sort(int a[],int n)

{

int i, j, temp;

for (i = 0; i < n - 1; i++)

for (j = 0; j < n – i - 1; j++)

if (a[j] > a[j + 1])

{

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

**s4.c**

#include <stdio.h>

int main()

{

extern void sortA(int a[], int asize);

extern void sortB(int b[], int bsize);

extern void sortAnB2C(int \*a, int asize, int \*b,

int bsize, int \*c, int csize);

int a[8], b[3];

int c[11];

int i;

for (i = 0; i < 8; i++)

scanf("%d", &a[i]);

for (i = 0; i < 3; i++)

scanf("%d", &b[i]);

sortA(a, 8);

sortB(b, 3);

sortAnB2C(a, 8, b, 3, c, 11);

return 0;

}

**sort.prj**

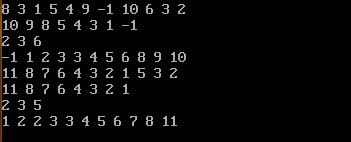
s1.c

s2.c

s3.c

s4.c

**运行结果：**



**总结：**s4.c中把a，b两个数组合并到c用的是归并排序，因为之前a数组是从大排到小，c数组要求是从小排到大，所以在函数中又写了一个冒泡来让a数组从小排到大，从而实现a，b的合并。

# 实验5

## 实验目的

1. 能够理解和使用指针变量；
2. 能够设计通过指针引用数组和字符串以及指向函数的指针的程序。
3. 能够定义和使用结构体类型变量；
4. 能够定义和使用结构体数组；
5. 初步掌握用指针处理链表的方法；
6. 了解共用体和枚举类型的概念。

## 实验工具

Tubro C 2.0/3.0

## 编写要求

对每个实验题目，需要提供如下内容：

1. 程序流程图（应采用NS图或传统流程图）；
2. 具体程序代码（具有良好的程序设计风格，如程序模块化、加注释、缩进的书写格式）；
3. 实验运行结果（对比测试用例）；
4. 对运行情况所做的分析以及调试程序所取得的经验收获。

## 排版要求

1. 标题四号宋体，正文小四宋体，段间距1.5倍；
2. 除封面以外，其他内容双面打印。

## 实验题目

1．ListN.prj -->ListN.exe

struct number {int c;struct number \*next};

T1.c-输入一串数字,均大于0，用空格隔开，把每一个数字数据存在一个单向动态链表（简称数字链表）里的一个节点里

struct number \* create\_number\_list( ) //生成链表

void print\_number\_list(struct number \* head) //访问输出链表的元素

T2.c-//在数字链表中查找指定的数字，找到则返回其是链表第几个节点值，否则返回0

int serach\_num(struct setn \*head，struct number )

T3.c

void main()

{

//用动态内存分配函数申请存放N个数字的空间，输入五个数字“9 1332 8765 128 234”

//调用函数create\_number\_list生成一个数字动态链表，分别存放了5个数字。

//调用函数print\_number\_list把数字动态链表的内容，分别输出。

//输入需要查找数字

//调用函数serach\_num,查找指定的数字 “1332”，输出查找的结果

//调用函数serach\_num,查找指定的数字 “128”，输出查找的结果

//调用函数serach\_num,查找指定的数字 “33”，输出查找的结果

}

1. 测试数据“9 1332 8765 128 234” ，先后查找“1332”和“128”和“33”

**T1.c——NS图**

struct number

{

long int num;

struct number \*next;

};

int n;

|  |  |  |
| --- | --- | --- |
| struct number \*head，\*p1，\*p2; | | |
| n=0 | | |
| 开辟一段新内存给p1,p2 | | |
| 输入p1->num | | |
| while(p1->num) | | |
|  | n++ | |
| N  n==1  Y | |
| head=p1 | p2->next=p1 |
| p2=p1 | |
| p1=(struct number\*)malloc(LEN) | |
| scanf("%ld",&p1->num) | |
| p2->next=NULL | | |
| return head; | | |

**T2.c---NS图**

|  |  |  |
| --- | --- | --- |
| struct number \*p=head | | |
| N  Y  head | | |
| while(p) | |  |
|  | printf("%ld\n",p->num)  p=p->next |

**T3.c—NS图**

|  |  |
| --- | --- |
| int m,k; | |
| struct number \*head | |
| head= create\_number\_list() | |
| print\_number\_list (head) | |
| while(scanf("%d",&m)==1) | |
|  | k=serach\_num(head,m) |
| printf("%d\n",k) |

int serach\_num(struct number \*head,int m)

|  |  |  |
| --- | --- | --- |
| struct number \*p=head; | | |
| n=1 | | |
| while (p) | | |
|  | N  m==p->num  Y | |
| return n |  |
| p=p->next  n++ | |
| return 0; | | |

**T1.c**

#include <stdio.h>

#include <stdlib.h>

#define LEN sizeof(struct number)

struct number

{

long int num;

struct number \*next;

};

int n;

struct number \*create\_number\_list ()

{

struct number \*head;

struct number \*p1, \*p2;

n = 0;

p1 = p2 = (struct number \*)malloc(LEN);

scanf("%ld", &p1->num);

head = NULL;

while (p1->num != 0)

{

n += 1;

if (n == 1)

head = p1;

else

p2->next = p1;

p2 = p1;

p1 = (struct number \*)malloc(LEN);

scanf("%ld", &p1->num);

}

p2->next = NULL;

return head;

}

**T2.c**

#include <stdio.h>

struct number

{

long int num;

struct number \*next;

};

void print\_number\_list(struct number \*head)

{

struct number \*p;

printf("\nNow,These %d records are :\n", 5);

p = head;

if (head != NULL)

do{

printf("%ld\n", p->num);

p = p->next;

}while (p != NULL);

printf("\n");

}

**T3.c**

#include <stdio.h>

struct number

{

long int num;

struct number \*next;

};

int main()

{

extern struct number \* create\_number\_list();

extern void print\_number\_list (struct number \*head);

int serach\_num(struct number \*head, int m);

int m, k;

struct number \*head;

head = create\_number\_list();

print\_number\_list (head);

printf("Please enter an integer number (q to quit): ");

while (scanf("%d", &m) == 1)

{

k = serach\_num(head, m);

printf("%d\n", k);

printf("Please enter next integer number (q to quit): ");

}

puts("Done.");

return 0;

}

int serach\_num(struct number \*head, int m)

{

struct number \*p = head;

int n = 1;

while (p)

{

if ( m == p->num)

return n;

p = p->next;

n++;

}

return 0;

}

**运行结果**

