# 

# 《计算机实践》

# 实验报告I—数据结构

班号：0321501 学号：032150115

姓名：易博天

Email：yiboxiaotian@qq.com

签名：

南京航空航天大学

## 实验1：线性表的实验

### 一、实验目的

　　掌握线性表的顺序存储结构和链式存储结构的相关操作实现。

### 二、实验内容

1、实验内容 顺序表的定义、创建、插入和删除操作实现。

（1） 初始化顺序表，并创建顺序表的初始序列；

（2） 在指定位置实现顺序表的插入操作，并输出插入操作后的序列；

（3） 在指定位置实现顺序表的删除操作，并输出删除操作后的序列。

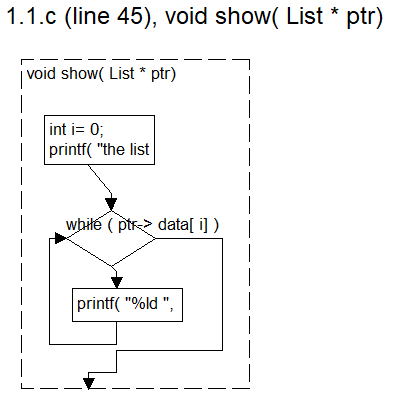
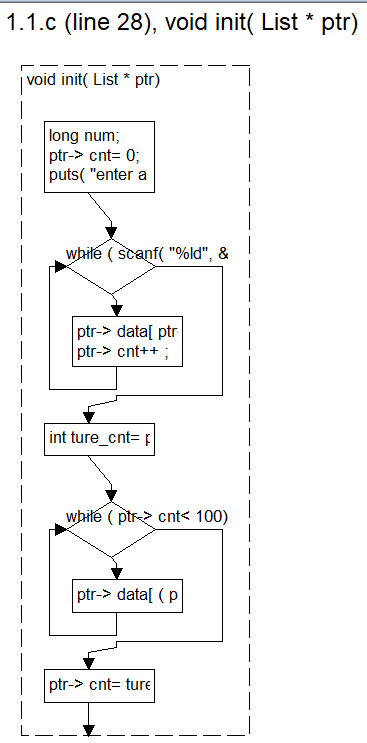
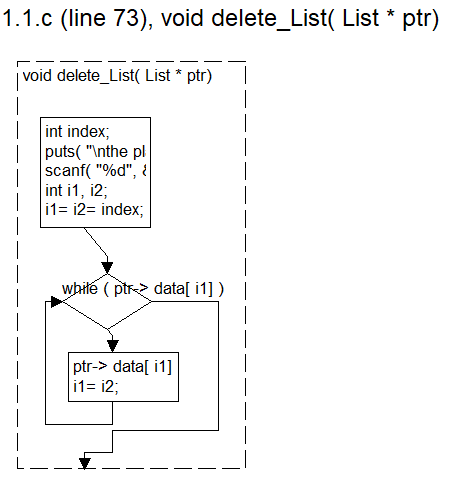
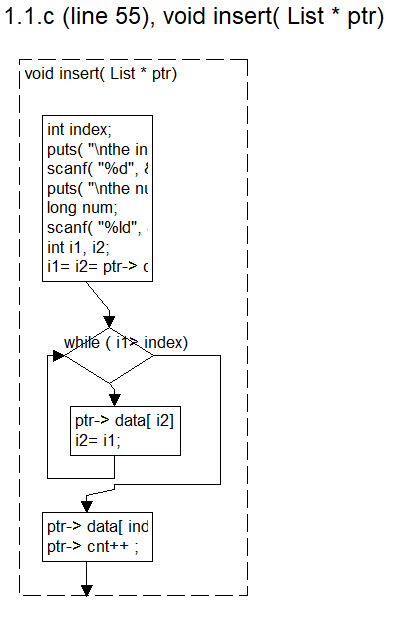
### 三、算法分析与程序设计

1. 输入数据：建立输入处理，输入顺序表里的数据（逻辑顺序预先确定，数据类型 为字符型或整型，数据值自行设置），插入和删除的位置要求从键盘输入。

2. 输出形式：分别输出顺序表的初始序列以及进行插入、删除操作以后的正确序列。

3. 算法描述

给出算法描述的程序流程图



4. 程序设计

给出实现程序功能的函数及说明，示例如下：

(1) 本程序包含的函数

主函数 main()

void init(List \*ptr);// 初始化函数，参数是顺序表地址，实现从键盘输入指定值初始化操作

void show(List \*ptr);// 显示顺序表，参数是顺序表地址

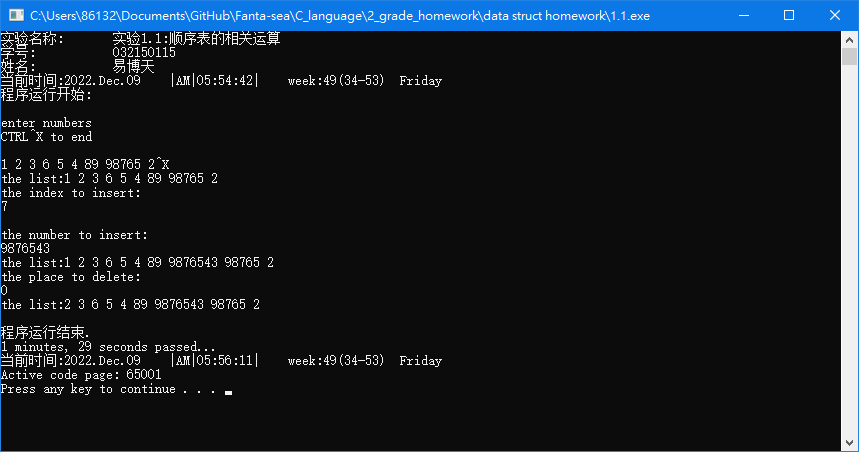
void insert(List \*ptr);// 向顺序表指定位置插入一个指定元素，参数是顺序表地址

void delete\_List(List \*ptr); // 从顺序表的指定位置删除元素，参数是顺序表地址

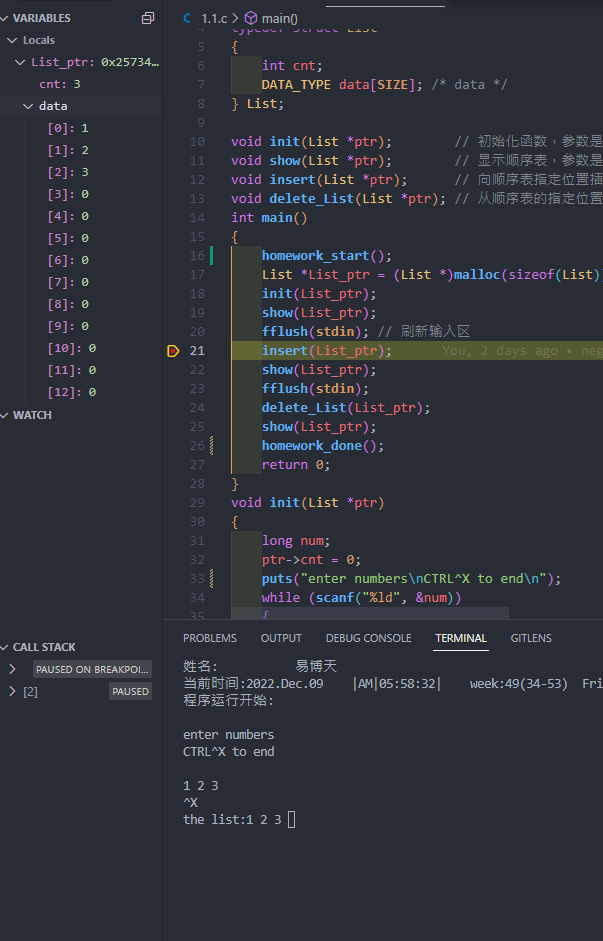
### 四、调试分析

#### 1、调试过程及说明

* 以截屏图像的方式给出，包括程序运行开始和结束时的系统时间、姓名、学号、实验名称等信息.



中间调试均采用打断点方式



#### 2、调试分析

1.在init函数中初始化将顺序表置零过程中因为重用cnt而导致cnt在总是等于数组大小，在加入true\_cnt暂存之后解决了这个问题。

2.通过刷新输入区函数fflush解决了scanf异常输入的问题

### 五、使用说明

此程序顺序表将0视为表结尾

因此实际上输入0即是删除和确定顺序表结尾

创建、插入、删除功能是顺序执行的，只是为了测试对应功能

**六、测试结果**

内容：给出测试用例测试程序。写出测试的情况，使用的输入以及得到的结果。

要求：测试的情况越全面越好。

示例：

**1 、建立单链表：**

输入1 23 987654 -56 -765432 166^X。得到顺序表（1 23 987654 -56 -765432 166）

**2 、插入：**

（1）测试插入位置不合法

输入-1，显示invalid input

（2）测试插入位置在头部

输入0 \n 100 ，得到顺序表（100 1 23 987654 -56 -765432 166）

（3）测试插入位置在尾部

输入6 \n 2，得到顺序表（1 23 987654 -56 -765432 166 2）

（4）测试插入位置在中间

输入3 \n 8，得到顺序表（1 23 987654 8 -56 -765432 166）

**3、删除：**

（1）测试删除位置不合法

输入 10 。返回invalid input

（2）测试删除位置在头部

输入 0 。得到顺序表（23 987654 -56 -765432 166）

（3）测试删除位置在中间

输入 4 。得到顺序表（1 23 987654 -56 166）

（4）测试删除位置在尾部

输入 5 。得到顺序表（1 23 987654 -56 -765432）

**七、实验的收获和感想**

只是简单的顺序表实现没必要整的那么花，虽然还有很多可以改进的地方，比如封装更好的输入函数和准备更多的错误情况，或者做一个选择菜单，以及顺序表定位和书上的一致。但是已经实现了主要功能，展现了数据结构，就不多下功夫了。

**八、附录：源代码（双栏排版，加必要的注释）**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

// 时间定义

static time\_t now\_time1, now\_time2; // 两个秒时间

static struct tm \*timeinfo; // 时间信息

static char time\_text[256]; // 时间信息文本

void homework\_start(void)

{

puts("实验名称:\t实验1.1:顺序表的相关运算");

puts("学号:\t\t032150115");

puts("姓名:\t\t易博天");

printf("当前时间:");

// 开始时间

time(&now\_time1);

timeinfo = localtime(&now\_time1);

strftime(time\_text, 80, "%Y.%b.%d |%p|%H:%M:%S| week:%W(34-53) %A", timeinfo); // 根据上述format显示时间信息文本

puts(time\_text);

// 开始时间戳完毕

puts("程序运行开始:\n");

}

void homework\_done(void)

{

puts("\n\n程序运行结束.");

time(&now\_time2);

timeinfo = localtime(&now\_time2);

strftime(time\_text, 80, "%Y.%b.%d |%p|%H:%M:%S| week:%W(34-53) %A", timeinfo);

printf("%d minutes, %d seconds passed...\n", (int)((now\_time2 - now\_time1) / 60), (now\_time2 - now\_time1) % 60);

printf("当前时间:");

puts(time\_text);

system("pause"); // 暂停以保留运行结果

}

#define SIZE 100

typedef long DATA\_TYPE;

typedef struct List

{

int cnt;

DATA\_TYPE data[SIZE]; /\* data \*/

} List;

void init(List \*ptr); // 初始化函数，参数是顺序表地址，实现从键盘输入指定值初始化操作

void show(List \*ptr); // 显示顺序表，参数是顺序表地址

void insert(List \*ptr); // 向顺序表指定位置插入一个指定元素，参数是顺序表地址

void delete\_List(List \*ptr); // 从顺序表的指定位置删除元素，参数是顺序表地址

int main()

{

homework\_start();

List \*List\_ptr = (List \*)malloc(sizeof(List));

init(List\_ptr);

show(List\_ptr);

fflush(stdin); // 刷新输入区

insert(List\_ptr);

show(List\_ptr);

fflush(stdin);

delete\_List(List\_ptr);

show(List\_ptr);

homework\_done();

return 0;

}

void init(List \*ptr)

{

long num;

ptr->cnt = 0;

puts("enter numbers\nCTRL^X to end\n");

while (scanf("%ld", &num))

{

ptr->data[ptr->cnt] = num;

ptr->cnt++;

} // enter some numbers

int true\_cnt = ptr->cnt;

while (ptr->cnt < SIZE)

{

ptr->data[(ptr->cnt)++] = 0;

} // other numbers become 0

ptr->cnt = true\_cnt;

}

void show(List \*ptr)

{

int i = 0;

printf("the list:");

while (ptr->data[i])

{

printf("%ld ", ptr->data[i++]);

}

// puts("\n");

}

void insert(List \*ptr)

{

int index;

puts("\nthe index to insert:");

scanf("%d", &index);

if (index < 0 || index > ptr->cnt)

{

puts("invalid input");

return;

}

puts("\nthe number to insert:");

long num;

scanf("%ld", &num);

int i1, i2;

i1 = i2 = ptr->cnt;

while (i1 > index)

{

ptr->data[i2] = ptr->data[--i1];

i2 = i1;

}

ptr->data[index] = num;

ptr->cnt++;

}

void delete\_List(List \*ptr)

{

int index;

puts("\nthe place to delete:");

scanf("%d", &index);

if (index < 0 || index > ptr->cnt)

{

puts("invalid input");

return;

}

int i1, i2;

i1 = i2 = index;

while (ptr->data[i1])

{

ptr->data[i1] = ptr->data[++i2];

i1 = i2;

}

}

# 实验 1.2 单链表的相关运算

### 二、实验内容

1、单链表的定义、创建及相关操作实现。

求先后输入的两个集合 A、B 的差集（A-B）和并集（A+B）。 说明：已知两个集合 A 和 B（每个集合内部元素各不相同），两个集合的差集（A-B） 中包含所有属于集合 A 而不属于集合 B 的元素；两个集合的并集（A+B）中包含所有属于 集合 A 和属于集合 B 的元素，且相同数据只存储一个。

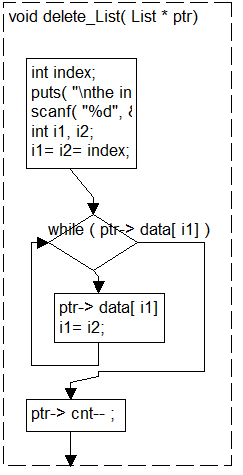
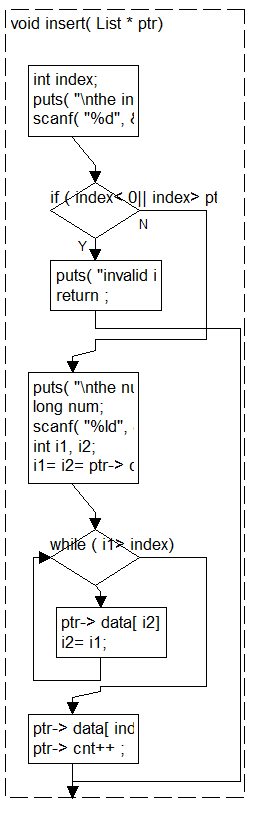
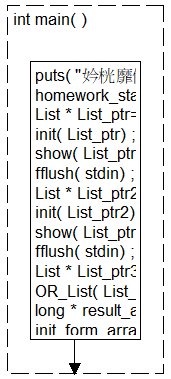
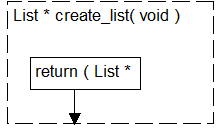
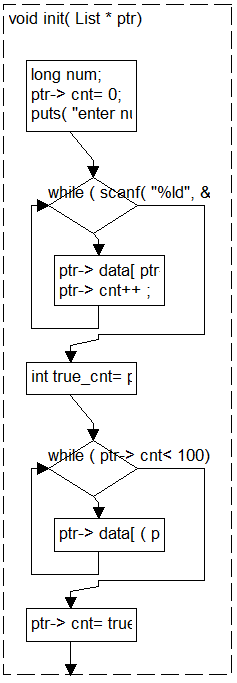
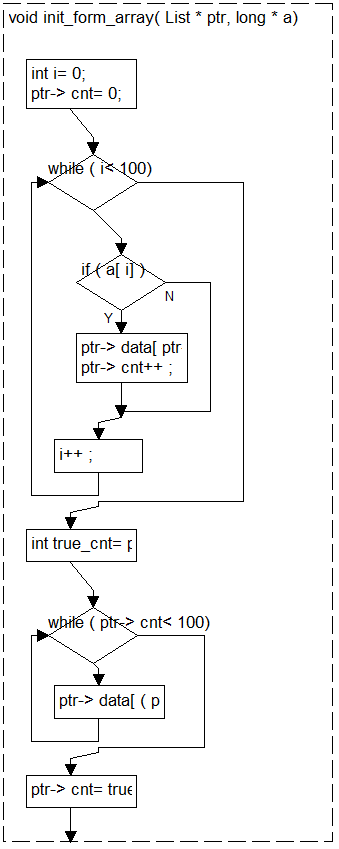
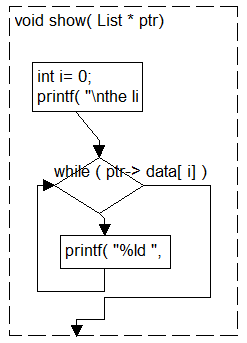
（1）集合 A、B 采用单链表的存储结构； （2）输出 A、B 原始序列、差集和并集。

三、算法分析与程序设计

（1）输入数据：建立输入处理，输入单链表 A 和 B 里的数据（逻辑顺序预先确定， 数据类型为字符型或整型，数据值自行设置）。

（2）输出形式：分别输出单链表 A 和 B 的原始序列、差集（A-B）和并集（A+B）。

3. 算法描述

给出算法描述的程序流程图

4. 程序设计

给出实现程序功能的函数及说明，示例如下：

(1) 本程序包含的函数

主函数 main()

void init(List \*ptr); // 初始化函数，参数是顺序表地址，实现从键盘输入指定值初始化操作

void show(List \*ptr); // 显示顺序表，参数是顺序表地址

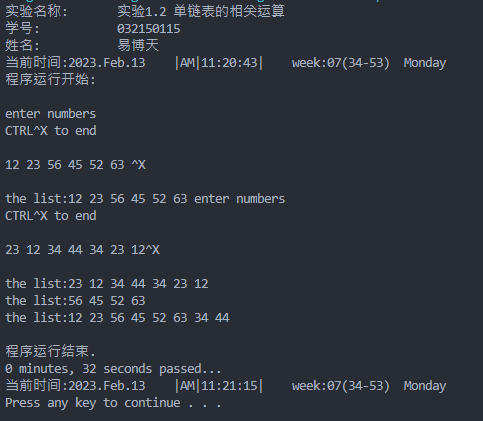
void insert(List \*ptr); // 向顺序表指定位置插入一个指定元素，参数是顺序表地址

void delete\_List(List \*ptr); // 从顺序表的指定位置删除元素，参数是顺序表地址

### 四、调试分析

#### 1、调试过程及说明

* 以截屏图像的方式给出，包括程序运行开始和结束时的系统时间、姓名、学号、实验名称等信息.



中间调试均采用打断点方式

#### 2、调试分析

1.在init函数中初始化将顺序表置零过程中因为重用cnt而导致cnt在总是等于数组大小，在加入true\_cnt暂存之后解决了这个问题。

2.通过刷新输入区函数fflush解决了scanf异常输入的问题

### 五、使用说明

此程序顺序表将0视为表结尾

因此实际上输入0即是删除和确定顺序表结尾

**六、测试结果**

内容：给出测试用例测试程序。写出测试的情况，使用的输入以及得到的结果。

要求：测试的情况越全面越好。

示例：如图所示

**七、实验的收获和感想**

只是简单的顺序表实现没必要整的那么花，虽然还有很多可以改进的地方，比如封装更好的输入函数和准备更多的错误情况，或者做一个选择菜单，以及顺序表定位和书上的一致。但是已经实现了主要功能，展现了数据结构，就不多下功夫了。

**八、附录：源代码（双栏排版，加必要的注释）**

#include "homework.h"

#define SIZE 100

typedef long DATA\_TYPE;

typedef struct List

{

int cnt;

DATA\_TYPE data[SIZE]; /\* data \*/

} List;

void init(List \*ptr); // 初始化函数，参数是顺序表地址，实现从键盘输入指定值初始化操作

void show(List \*ptr); // 显示顺序表，参数是顺序表地址

void insert(List \*ptr); // 向顺序表指定位置插入一个指定元素，参数是顺序表地址

void delete\_List(List \*ptr); // 从顺序表的指定位置删除元素，参数是顺序表地址

List \*create\_list(void); // 创建新顺序表

long \*OR\_List(List \*List\_ptr, List \*List\_ptr2); // 试试输出到缓存区是否可行

void init\_form\_array(List \*ptr, long \*a); // as the name

long \*AND\_List(List \*List\_ptr, List \*List\_ptr2);

int main(void)

{

puts("实验名称: 实验1.2 单链表的相关运算");

homework\_start();

List \*List\_ptr = create\_list(); // one

init(List\_ptr);

show(List\_ptr);

fflush(stdin); // 刷新输入区

// insert(List\_ptr);

// show(List\_ptr);

// fflush(stdin);

// delete\_List(List\_ptr);

// show(List\_ptr);

List \*List\_ptr2 = create\_list(); // another

init(List\_ptr2);

show(List\_ptr2);

fflush(stdin); // 刷新输入区

List \*List\_ptr3 = create\_list(); // result

OR\_List(List\_ptr, List\_ptr2); // 能否通过输入到缓存区的方式初始化顺序表？

long \*result\_array = AND\_List(List\_ptr, List\_ptr2);

init\_form\_array(List\_ptr3, result\_array);

show(List\_ptr3);

result\_array = OR\_List(List\_ptr, List\_ptr2);

init\_form\_array(List\_ptr3, result\_array);

show(List\_ptr3);

homework\_done();

return 0;

}

List \*create\_list(void)

{

return (List \*)malloc(sizeof(List));

}

void init(List \*ptr)

{

long num;

ptr->cnt = 0;

puts("enter numbers\nCTRL^X to end\n");

while (scanf("%ld", &num))

{

ptr->data[ptr->cnt] = num;

ptr->cnt++;

} // enter some numbers

int true\_cnt = ptr->cnt;

while (ptr->cnt < SIZE)

{

ptr->data[(ptr->cnt)++] = 0;

} // other numbers become 0

ptr->cnt = true\_cnt;

}

void init\_form\_array(List \*ptr, long \*a)

{

int i = 0;

ptr->cnt = 0;

while (i < SIZE)

{

if (a[i])

{

ptr->data[ptr->cnt] = a[i];

ptr->cnt++;

}

i++;

} // enter some numbers

int true\_cnt = ptr->cnt;

while (ptr->cnt < SIZE)

{

ptr->data[(ptr->cnt)++] = 0;

} // other numbers become 0

ptr->cnt = true\_cnt;

}

void show(List \*ptr)

{

int i = 0;

printf("\nthe list:");

while (ptr->data[i])

{

printf("%ld ", ptr->data[i++]);

}

// puts("\n");

}

void insert(List \*ptr)

{

int index;

puts("\nthe index to insert:");

scanf("%d", &index);

puts("\nthe number to insert:");

long num;

scanf("%ld", &num);

int i1, i2;

i1 = i2 = ptr->cnt;

while (i1 > index)

{

ptr->data[i2] = ptr->data[--i1];

i2 = i1;

}

ptr->data[index] = num;

ptr->cnt++;

}

void delete\_List(List \*ptr)

{

int index;

puts("\nthe index to delete:");

scanf("%d", &index);

int i1, i2;

i1 = i2 = index;

while (ptr->data[i1])

{

ptr->data[i1] = ptr->data[++i2];

i1 = i2;

}

ptr->cnt--;

}

long \*OR\_List(List \*List\_ptr, List \*List\_ptr2) // 试试输出到缓存区是否可行

{

// int i2 = 0;

int k = 0;

static long a[SIZE] = {0};

// while ((List\_ptr->data[i]) || (List\_ptr2->data[i]))

//{

/\*while ((List\_ptr->data[i]) || (List\_ptr2->data[i]))

{

int i2=i;

}

if ((List\_ptr->data[i]) == (List\_ptr2->data[i]))

{

a[k++] = List\_ptr->data[i++];

}

else

{

a[k++] = List\_ptr->data[i];

a[k++] = List\_ptr2->data[i++];

}

long key =List\_ptr->data[i];\*/

for (int i = 0; i < (List\_ptr->cnt); i++)

a[k++] = List\_ptr->data[i];

for (int i = 0; i < (List\_ptr2->cnt); i++)

a[k++] = List\_ptr2->data[i];

for (int i = 0; i < ((List\_ptr2->cnt) + (List\_ptr->cnt));)

{

long key = a[i];

if (a[i])

{

for (int in = i + 1; in < ((List\_ptr2->cnt) + (List\_ptr->cnt)); in++)

{

if (key == a[in])

a[in] = 0;

}

}

i++;

}

// 抓住一个数，分别在两个数组中捋一遍，重复此过程\_被否定

// 全部写入一个数组，然后查重

//}

return a;

}

// 相同的都归0，只要第一组

long \*AND\_List(List \*List\_ptr, List \*List\_ptr2)

{

// int i2 = 0;

int k = 0;

static long a[SIZE] = {0};

for (int i = 0; i < (List\_ptr->cnt); i++)

a[k++] = List\_ptr->data[i];

for (int i = 0; i < (List\_ptr2->cnt); i++)

a[k++] = List\_ptr2->data[i];

for (int i = 0; i < ((List\_ptr2->cnt) + (List\_ptr->cnt));)

{

long key = a[i];

if (a[i])

{

for (int in = i + 1; in < ((List\_ptr2->cnt) + (List\_ptr->cnt)); in++)

{

if (key == a[in])

a[i] = a[in] = 0;

}

}

i++;

}

for (int i = List\_ptr->cnt; i < ((List\_ptr2->cnt) + (List\_ptr->cnt)); i++)

a[i] = 0;

return a;

}