实验四

1. 实验目的和要求

数组与指针 【实验目的】

1、进一步加深对数组的理解，掌握数组的定义方法；

2、掌握数组的处理方法、数组作为函数参数的使用方法，以及搜索与排序的应用。

3、掌握指针的概念、指针变量定义格式以及指针的运算；

4、掌握指针与数组、函数的关系；

5、理解内存动态分配的含义、熟练掌握内存动态分配方法；

1. 掌握递归函数的定义方法

**二、实验内容**

1. 数组
2. 打印不同的数： 编写一个程序，读入 10 个数，输出其中不同的数（即如果一个数出现多次，只打印一次）。 提示：读入的数如果是一个新的值，则将其存入一个数组。否则，将其丢弃。输入完毕后， 数组中保存的就是不同的数。 下面是一个运行样例： Enter ten numbers: 1 2 3 2 1 6 3 4 5 2 The distinct numbers are: 1 2 3 6 4 5
3. 起泡排序： 利用起泡排序算法编写一个排序函数。起泡排序算法分若干趟对数组进行处理。每趟处理中， 对相邻元素进行比较。若为降序，则交换；否则，保持原顺序。此技术被称为起泡排序（bubble sort）或下沉排序（sinking sort），因为较小的值逐渐地“冒泡”到上部，而较大值逐渐下沉到底 部。 算法可描述如下： bool changed = true; do { changed = false; for (int j = 0; j < listSize – 1; j++) if (list[j] > list[j+1]) { swap list[j] with list[j+1]; changed = true; } } while (changed); 很明显，循环结束后，列表变为升序。容易证明 do 循环最多执行 listSize – 1 次。 编写测试程序，读入一个含有 10 个双精度数字的数组，调用函数并显示排列后的数字。

3.游戏：存物柜问题： 一个学校有 100 个存物柜，100 个学生。开学第一天所有存物柜都是关闭的。第一个学生（记 为 S1）来到学校后，打开所有的存物柜。第二个学生 S2，从第二个存物柜（记为 L2）开始，每 隔两个存物柜，将它们关闭。第三个学生 S3 从第三个存物柜 L3 开始，每隔三个，将它们的状态 Enter 10 改变（开着的关上，关着的打开）。学生 S4，从 L4 开始，每隔四个改变它们的状态。学生 S5， 从 L5 开始，每隔五个改变状态。依此类推，直至学生 S100 改变 L100 的状态。 当所有学生完成这个过程，那些存物柜是开着的？编写一个程序求解此问题，显示所有开着 的柜子号码，号码之间用一个空格隔开。 提示：使用一个 100 个布尔型元素的数组，每个元素代表存物柜是开（true）或关（false）。 最初所有的储物柜都是关闭的

4、合并两个排列好的数组： 编写如下函数，合并两个排列好的数组，形成一个新的排列好的数组。 void merge(const int list1[], int size1, const int list2[], int size2, int list3[]) 使用 size1+size2 次比较实现函数。编写测试程序，提示用户输入两个排列好的数组，并显示 合并以后的数组。下面是一个运行样例。注意，输入数据的第一个数字是数组的元素数，而不是 数组的一部分。假定数组大小不超过 80。 Enter list1: 5 1 5 16 61 111 Enter list1: 4 2 4 5 6 The merged list is 1 2

5、检验子串： 编写如下函数，检验 C 字符串 s1 是否是 C 字符串 s2 的子串。如果匹配，返回 s1 在 s2 中的 下标，否则返回–1。 int indexOf(const char s1[], const char s2[]) 编写测试程序，读入两个 C 字符串，检验 C 字符串 s1 是否是 C 字符串 s2 的子串。下面是程 序的运行样例： Enter the first string: welcome Enter the second string: We welcome you! indexOf(“welcome”, “We welcome you!”) is 3 Enter the first string: welcome Enter the second string: We invite you! indexOf(“welcome”, “We invite you!”) is –1

6. 字符串中每个字母出现的次数： 请使用如下函数头编写函数，数出字符串中每个字母出现的次数。 void count(const char s[], int counts[]) counts 是一个有 26 个元素的整数数组。const[0]，const[1]，…，const[25]分别记录 a，b，…， z 出现的次数。字母不分大小写，例如字母 A 和字母 a 都被看作 a。 编写测试程序，读入字符串并调用 count 函数，显示非零的次数。下面是程序的一个运行样 例：

Enter Enter Enter Enter Enter Enter Enter 11 Enter a string: Welcome to New York!

c: 1 times

e: 3 times

k: 1 times

l: 1 times

m: 1 times

n: 1 times

o: 3 times

r: 1 times

t: 1 times

w: 2 times y: 1

（二）指针 1、上机验证下列程序的运行结果（有错误的话自己补充完善）

(1) void main() ｛ int i,j,\*pi,\*pj; //此处的\*表示定义指针变量，而非间接运算符 pi=&i; pj=&j; i=5;j=7; cout<<i<<’\t’<<j<<’\t’<<pi<<’\t’<<<&i<<’\t’<<\*&i<<’\t’<<&j<<’\t’<<\*&j; } 运行结果： 上述结果中，pi 与&i,pj 与&j 是地址值，随编译程序而变化，不确定

(2) int main() //C 语言程序，要了解

{

int a[]={1,2,3};

int \*p,i;

p=a; //将数组 a 首地址送给 p

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

printf("%d,%d,%d,%d\n",a[i],p[i],\*(p+i),\*(a+i)); //与 cout 功能差不多

}

运行结果：

1,1,1,1

2,2,2,2

3,3,3,3

通过这两道题目，希望学生掌握数组元素与指向数组的指针的不同。

a[i]表示数组中下标为 i 的元素。

a[i]←p[i]←\*(p+i)←\*(a+i)

a 是数组名，表示数组首地址，(p+i)表示数组中第 i 个元素的地址，\*(p+i)

(3)通过如下的问题理解递归函数的定义与调用（递归未讲，可以后做）

//#include “stdio.h” void f(char \*st,int i)

{

12

st[i]=’\0’;

cout<<st; // printf(“%s\n”,st);

if (i>1) f(st,i-1);

}

void main()

{

char st[]=”abcd”;

f(st,4);

}

补充完整，运行时输出为\_\_\_\_\_\_\_

(4)下面程序的主函数中能保证 p[0]输出 1，p[1]输出 2 吗？如何修改以保证之（提示：在函

数 f 中使用 new 生成动态数组；在 main 中用 delete 释放。）

#include<iostream>

using namespace std;

int \*f()

{

int list[]={1,2,3,4};

return list;

}

void main()

{

int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

2、程序设计

(1)编写函数检查字符串 s1 是否为字符串 s2 的子串，若是，返回第一次匹配的下标，否则返

回-1。在主程序中输入字符串 s1 与 s2，调用函数实现。

函数原型：int indexof(const char \*s1,const char \*s

2)编写一个函数将以字符串形式表示的一个 16 进制数转换为 10 进制数，并在主函数中测

试。函数原型 int parseHex(const char \*const hexString);

如：调用函数 parseHex(“A5”);返回 16

(3) 主程序中建立一动态数组（使用 new），数组元素及元素个数由键盘输入，动态调试观

察指针及指针指向的内容；设计一个函数对数组由小到大排序；主程序中用指针方式输出数组元

素；最后释放数组内存（delete）

**三、实验步骤、算法与结果分析**

**项目1、**

#include<iostream>

using namespace std;

int main()

{

int a[10];

int t = 1;

for (int b = 0; b <= 9; b++)

{

cin >> a[b];

}

for(int i=0;i<=9;i++,t=1)

{

for (int j=0; j <i; j++)

{

if (a[i] == a[j])

{

t = 0;

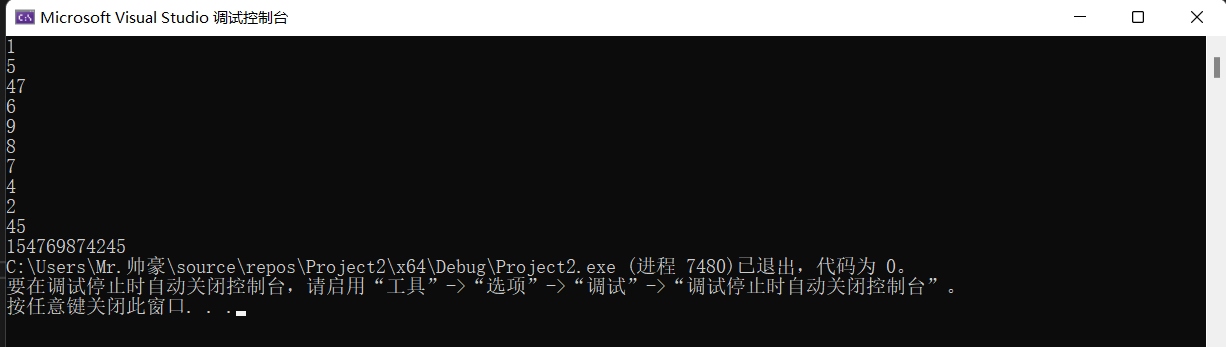
}

}

if (t == 1)

cout << a[i];

}



}

**项目2、**

#include<iostream>

#include<iomanip>

using namespace std;

const double eps = 1e-6;

double a[10];

void swap(double & b,double & c)

{

if (b-c>eps)

{

double temp=b;

b = c;

c = temp;

}

}

int main()

{

for (int i=0;i<=9;i++)

{

cin >> a[i];

}

for (int m = 0; m < 9; m++)

{

for (int j = 0; j <9-m; j++)

{

swap(a[j], a[j + 1]);

}

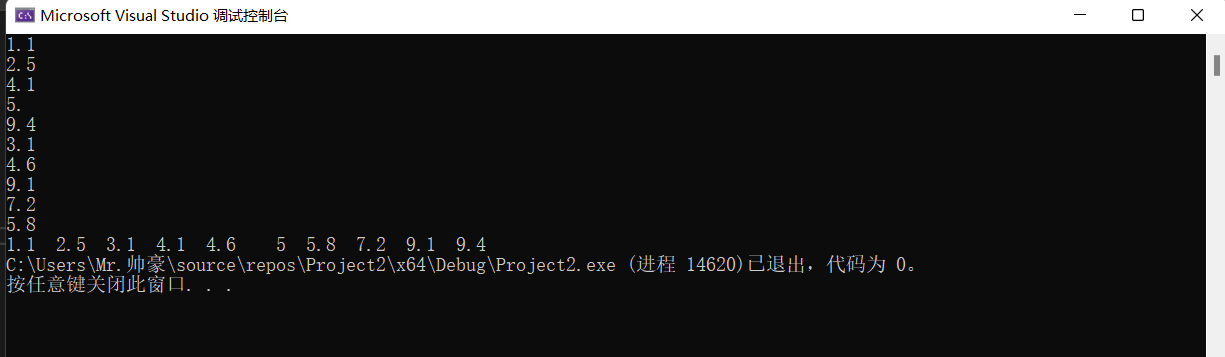
}

for (int k = 0; k <= 9; k++)

{

cout << a[k]<<setw(5);

}

}

**项目3、**

#include<iostream>

using namespace std;

int main()

{

int a[100];

for (int n=0;n<100;n++)

{

a[n] = 1;

}

for (int i=1;i<=100;i+2)

{

a[i] = 0;

}

for (int m = 3; m <= 100; m++)

{

for(int t=m-1;t<=99;t=t+m)

{

if (a[m] = 1)

{

a[m] = 0;

}

else if (a[m] = 0)

{

a[m] = 1;

}

}

}

for (int f = 0; f < 99; f++)

{

if (a[f] = 1)

{

cout << f+1 << endl;

}

}

}

****

**项目4、**

#include<iostream>

using namespace std;

void merge(const int list1[], int size1, const int list2[], int size2, int list3[])

{

for (int na = 0; na < size1; na++)

{

list3[na] = list1[na];

}

for (int nb = 0; nb < size2; nb++)

{

list3[nb + size1] = list2[nb];

}

int zj;

for (int n = 0; n < size1 + size2 - 1; n++)

{

int m = n + 1;

for (; m < size1 + size2; m++)

{

if (list3[n] > list3[m])

{

zj = list3[n];

list3[n] = list3[m];

list3[m] = zj;

}

}

}

}

int main()

{

int sz1, sz2, sz3;

cout << "Enter list1:";

cin >> sz1;

int\* ar1 = new int[sz1];

for (int nm = 0; nm < sz1; nm++)

{

cin >> ar1[nm];

}

cout << "Enter list2:";

cin >> sz2;

int\* ar2 = new int[sz2];

for (int mn = 0; mn < sz2; mn++)

{

cin >> ar2[mn];

}

sz3 = sz1 + sz2;

int\* ar3 = new int[sz3];

merge(ar1, sz1, ar2, sz2, ar3);

cout << "The merged list is ";

for (int r = 0; r < sz3; r++)

{

cout << ar3[r] << " ";

}

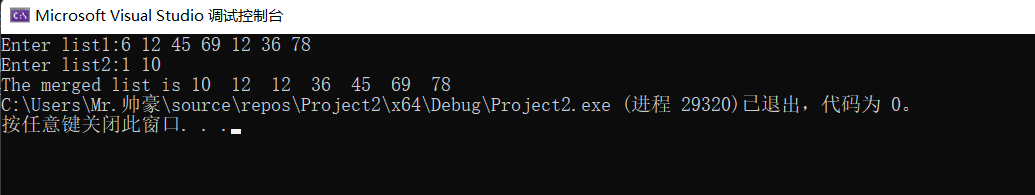
delete[]ar1;

delete[]ar2;

delete[]ar3;

return 0;

}

****

**项目5、**

#include<iostream>

using namespace std;

static int mm1 = 0, mm2 = 0;

int indexOf(const char s1[], const char s2[])

{

int n1 = mm1 + 1;

int n2 = mm2 + 1;

int b = -1;

int f = 0;

for (int m2 = 0; m2 < n2; m2++)

{

int c = 0;

if (s1[0] == s2[m2])

{

b = 1;

if (b != -1)

{

int m1 = 0;

for (; m1 < n1; m1++)

{

if (s1[m1] == s2[m1 + m2])

{

c++;

}

}

}

}

if (c == n1)

{

f++;

return m2; break;

}

}if (f == 0)

{

return -1;

}

}

int main()

{

char ss1[100], ss2[100];

cout << "Enter the first string:";

char arr1, arr2;

while ((arr1 = getchar()) != '\n')

{

ss1[mm1] = arr1;

mm1++;

}

cout << "Enter the second string:";

while ((arr2 = getchar()) != '\n')

{

ss2[mm2] = arr2;

mm2++;

}

cout << "indexOf(ss1, ss2) is " << indexOf(ss1, ss2);

return 0;

}

****

**六**

#include<iostream>

using namespace std;

static int mm1 = 0;

void count(const char s[], int counts[])

{

for (int n = 0; n <= mm1; n++)

{

if (s[n] == 'a' || s[n] == 'A')

{

counts[0]++;

}

else if (s[n] == 'b' || s[n] == 'B')

{

counts[1]++;

}

else if (s[n] == 'c' || s[n] == 'C')

{

counts[2]++;

}

else if (s[n] == 'd' || s[n] == 'D')

{

counts[3]++;

}

else if (s[n] == 'e' || s[n] == 'E')

{

counts[4]++;

}

else if (s[n] == 'f' || s[n] == 'F')

{

counts[5]++;

}

else if (s[n] == 'g' || s[n] == 'G')

{

counts[6]++;

}

else if (s[n] == 'h' || s[n] == 'H')

{

counts[7]++;

}

else if (s[n] == 'i' || s[n] == 'I')

{

counts[8]++;

}

else if (s[n] == 'j' || s[n] == 'J')

{

counts[9]++;

}

else if (s[n] == 'k' || s[n] == 'K')

{

counts[10]++;

}

else if (s[n] == 'l' || s[n] == 'L')

{

counts[11]++;

}

else if (s[n] == 'm' || s[n] == 'M')

{

counts[12]++;

}

else if (s[n] == 'n' || s[n] == 'N')

{

counts[13]++;

}

else if (s[n] == 'o' || s[n] == 'O')

{

counts[14]++;

}

else if (s[n] == 'p' || s[n] == 'P')

{

counts[15]++;

}

else if (s[n] == 'q' || s[n] == 'Q')

{

counts[16]++;

}

else if (s[n] == 'r' || s[n] == 'R')

{

counts[17]++;

}

else if (s[n] == 's' || s[n] == 'S')

{

counts[18]++;

}

else if (s[n] == 't' || s[n] == 'T')

{

counts[19]++;

}

else if (s[n] == 'u' || s[n] == 'U')

{

counts[20]++;

}

else if (s[n] == 'v' || s[n] == 'V')

{

counts[21]++;

}

else if (s[n] == 'w' || s[n] == 'W')

{

counts[22]++;

}

else if (s[n] == 'x' || s[n] == 'X')

{

counts[23]++;

}

else if (s[n] == 'y' || s[n] == 'Y')

{

counts[24]++;

}

else if (s[n] == 'z' || s[n] == 'Z')

{

counts[25]++;

}

}

}

int main()

{

char ss1[100];

cout << "Enter a string:";

char arr1;

while ((arr1 = getchar()) != '\n')

{

ss1[mm1] = arr1;

mm1++;

}

int ar[26] = { 0 };

char arr[26] = { 'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z' };

count(ss1, ar);

for (int m = 0; m < 26; m++)

{

if (ar[m] != 0)

{

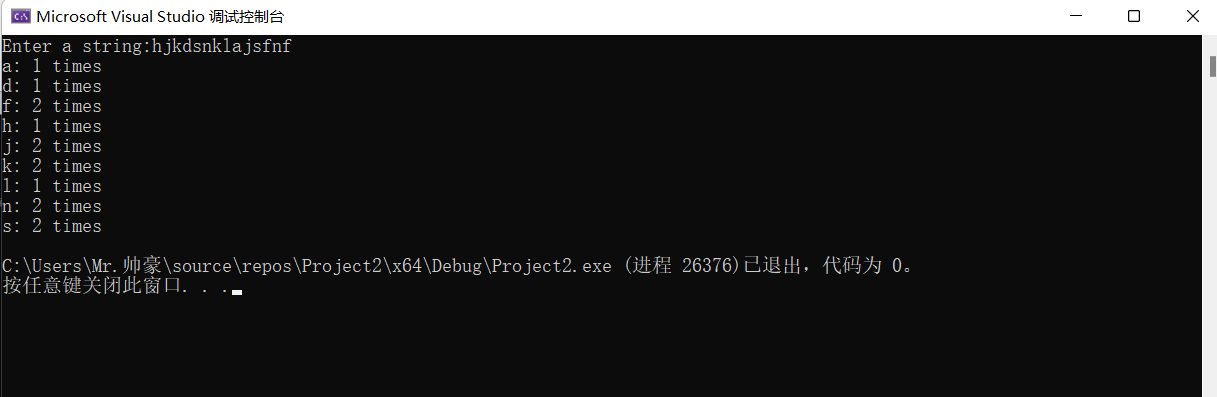
cout << arr[m] << ": " << ar[m] << " " << "times" << endl;

}

}

return 0;

}

****

**(二)**

**2**

#include<iostream>

using namespace std;

static int mm1 = 0, mm2 = 0;

int indexOf(const char\* s1, const char\* s2)

{

int n1 = mm1 + 1;

int n2 = mm2 + 1;

int b = -1;

int f = 0;

for (int m2 = 0; m2 < n2; m2++)

{

int c = 0;

if (s1[0] == s2[m2])

{

b = 1;

if (b != -1)

{

int m1 = 0;

for (; m1 < n1; m1++)

{

if (s1[m1] == s2[m1 + m2])

{

c++;

}

}

}

}

if (c == n1)

{

f++;

return m2; break;

}

}if (f == 0)

{

return -1;

}

}

int main()

{

char ss1[100], ss2[100];

cout << "Enter the first string:";

char arr1, arr2;

while ((arr1 = getchar()) != '\n')

{

ss1[mm1] = arr1;

mm1++;

}

cout << "Enter the second string:";

while ((arr2 = getchar()) != '\n')

{

ss2[mm2] = arr2;

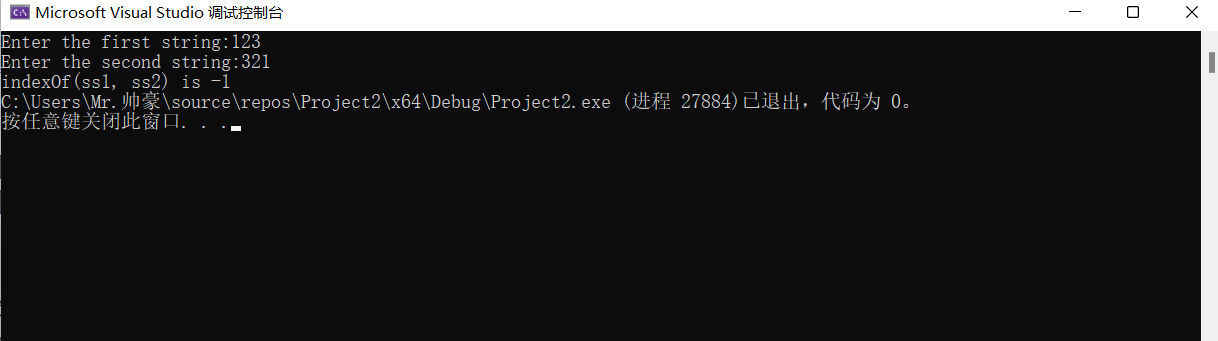
mm2++;

}

cout << "indexOf(ss1, ss2) is " << indexOf(ss1, ss2);

return 0;

}



**}**

**3.**

#include<iostream>

using namespace std;

int parseHex(const char\* const hexString)

{

int n = 0;

int m = strlen(hexString);

int z;

for (int x = 0, y = m - 1; x < m; y--, x++)

{

if (hexString[x] == 'A')

{

z = 10;

n += z \* pow(16, y);

}

else if (hexString[x] == 'B')

{

z = 11;

n += z \* pow(16, y);

}

else if (hexString[x] == 'C')

{

z = 12;

n += z \* pow(16, y);

}

else if (hexString[x] == 'D')

{

z = 13;

n += z \* pow(16, y);

}

else if (hexString[x] == 'E')

{

z = 14;

n += z \* pow(16, y);

}

else if (hexString[x] == 'F')

{

z = 15;

n += z \* pow(16, y);

}

else {

z = hexString[x] - 48;

n += z \* pow(16, y);

}

}

return n;

}

int main()

{

char arr[100];

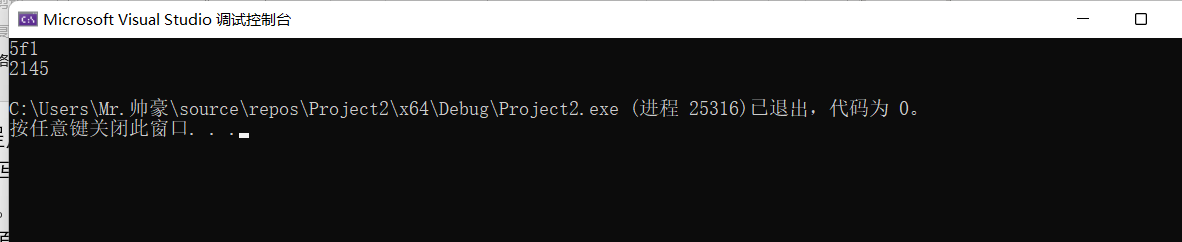
cin >> arr;

char ar1 = '5';

cout << parseHex(arr) << endl;

return 0;

}

****

**4.**

#include<iostream>

using namespace std;

void px(int list[], int n1)

{

int zj;

for (int n = 0; n < n1-1; n++)

{

int m = n + 1;

for (; m < n1; m++)

{

if (list[n] > list[m])

{

zj = list[n];

list[n] = list[m];

list[m] = zj;

}

}

}

}

int main()

{

int sz1;

cout << "Enter list:";

cin >> sz1;

int\* ar1 = new int[sz1];

for (int nm = 0; nm < sz1; nm++)

{

cin >> ar1[nm];

}

px(ar1,sz1);

for (int r = 0; r < sz1; r++)

{

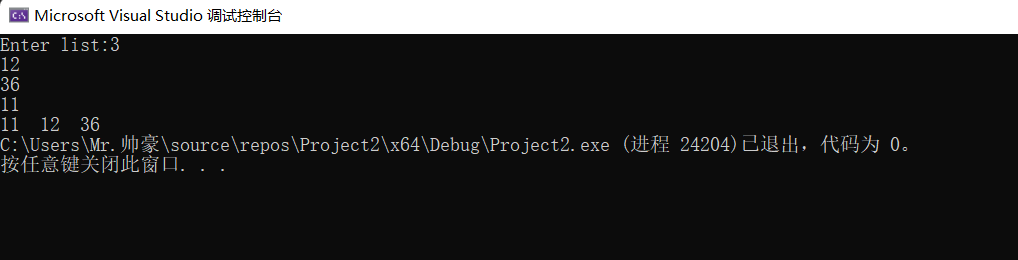
cout << \*(ar1+r) << " ";

}

delete[]ar1;

return 0;

}

****

**四、遇到的问题和解决办法**

**感觉最大的问题还是打代码打的太少了,很多时候一些简单的循环要修修改改好多次才行,还有就是对问题的数学分析还不到位,有时甚至给出错误的解决方案.我觉得有必要加强代码的编写.**

**五.体会**

**最大的体会是c++要多敲代码,不能拘泥于课本,需要更多的实践.**