**计算机程序设计基础（C++）**

**实验报告**

专业班级 软件工程2206班

学 号

姓 名

**实验报告成绩：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |

**批阅老师:\_\_\_\_\_\_\_\_**

**实验四 指针与数组**

1. **实验目的与要求**

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；

6、掌握递归函数的定义方法。

1. **实验内容**

(一)数组

1、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:

2.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（bubblesort）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 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 times

（二）指针

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 \*s2);

(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 165

(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）。

1. **实验步骤、算法与结果分析**
2. **数组**

**第一题：**

**Code：**

#include<iostream>

using namespace std;

bool is\_exist(int n, int a[],int num)

{

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

{

if (n == a[i])

{

return true;

break;

}

}

return false;

}

int main()

{

const int maxn = 10;

int a[maxn],t=0;

cout << "Enter ten numbers:";

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

{

int n;

cin >> n;

if (is\_exist(n, a,t) == 0)

{

a[t] = n;

t++;

}

}

cout << "The distinct numbers are:";

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

{

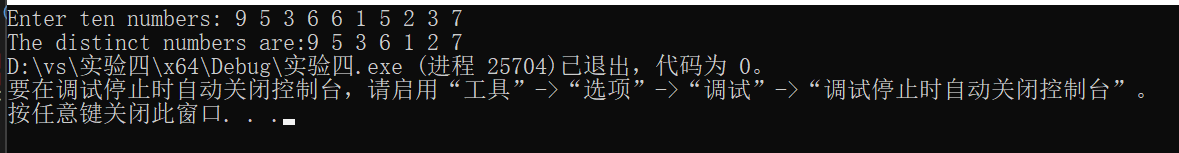
cout << a[i] << " ";

}

return 0;

}

**Result：**

****

**第二题：**

**Code：**

#include<iostream>

using namespace std;

void sort(double list[],int listsize)

{

bool changed = true;

do

{

changed = false;

for (int j = 0; j < listsize - 1; j++)

{

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

{

double t = list[j];

list[j] = list[j + 1];

list[j + 1] = t;

changed = true;

}

}

} while (changed);

}

int main()

{

const int maxn = 10;

double list[maxn];

cout << "please enter ten numbers";

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

{

cin >> list[i];

}

sort(list, maxn);

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

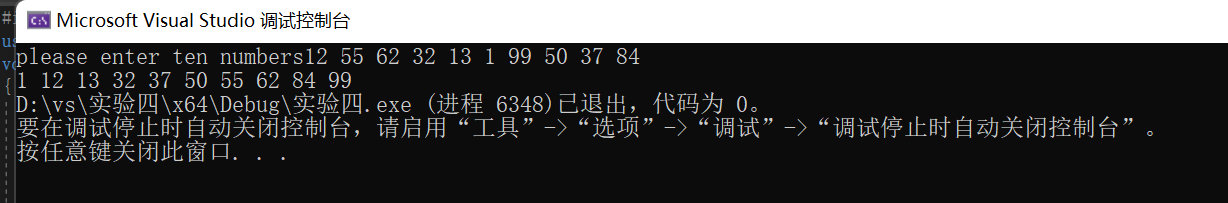
{

cout << list[i] << " ";

}

}

**Result：**

****

**第三题：**

**Code：**

#include<iostream>

using namespace std;

void o\_c(bool sum[],int n,int maxn)

{

for (int i = n-1; i < maxn; i += n)

{

sum[i]=!sum[i];

}

}

int main()

{

const int maxn = 100;

bool sum[maxn] = { 0 };

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

{

o\_c(sum, i, maxn);

}

cout << "开着的柜子为";

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

{

if (sum[i])

{

cout << i + 1 << " ";

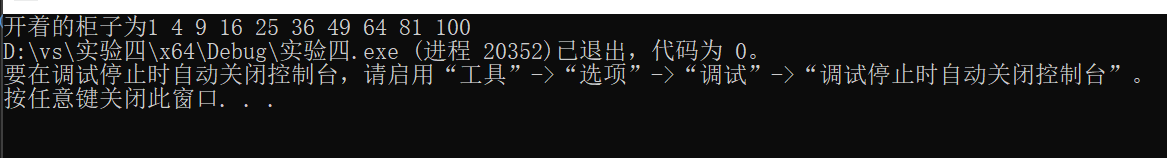
}

}

return 0;

}

**Result：**

****

**第四题：**

**Code：**

#include<iostream>

using namespace std;

void sort(int list[], int listsize)

{

bool changed = true;

do

{

changed = false;

for (int j = 0; j < listsize - 1; j++)

{

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

{

int t = list[j];

list[j] = list[j + 1];

list[j + 1] = t;

changed = true;

}

}

} while (changed);

}

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

{

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

{

list3[i] = list1[i];

}

for (int i = size1; i < size1 + size2; i++)

{

list3[i] = list2[i - size1];

}

sort(list3, size1 + size2);

}

int main()

{

const int maxn = 80;

int list1[maxn], list2[maxn], list3[maxn \* 2],size1,size2;

cout << "Enter list1:";

cin >> size1;

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

{

cin >> list1[i];

}

cout << "Enter list2:";

cin >> size2;

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

{

cin >> list2[i];

}

merge(list1, size1, list2, size2, list3);

cout << "The merged list is:";

for (int i = 0; i < size1 + size2; i++)

{

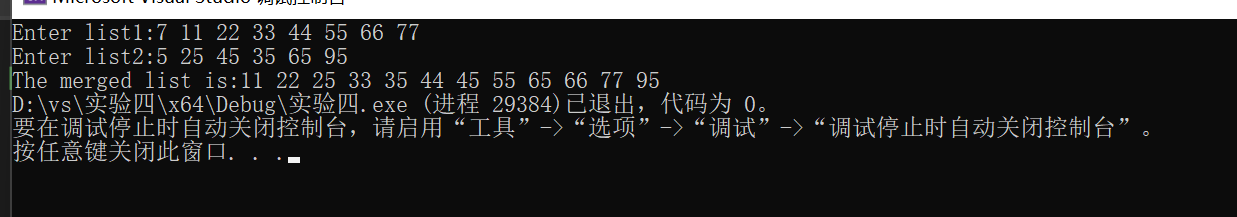
cout << list3[i] << " ";

}

return 0;

}

**Result：**

****

**第五题：**

**Code：**

#include<iostream>

using namespace std;

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

{

int len1 = strlen(s1), len2 = strlen(s2);

for (int i = 0; i <= len2 - len1; i++)//找第一个相同点

{

if (s1[0] == s2[i])//成功找到初始相同点

{

for (int j = 0; j < len1; j++)

{

if (s1[j] != s2[j + i])//发现不同

{

break;

}

if (j == len1 - 1)//全部核对

{

return i;

}

}

}

}

return -1;//第一个相同点未找到

}

int main()

{

const int maxn = 100;

char s1[maxn], s2[maxn];

cout << "Enter the first string:";

cin.getline(s1, maxn);

cout << "Enter the second string:";;

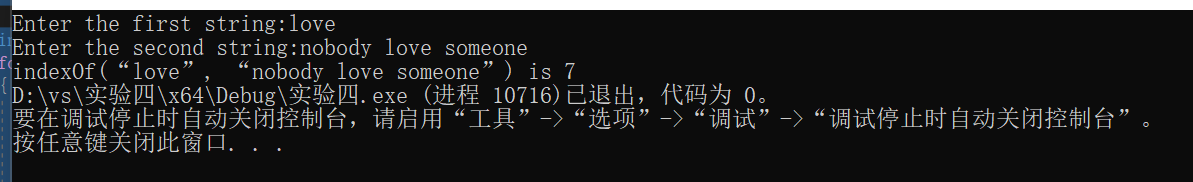
cin.getline(s2, maxn);

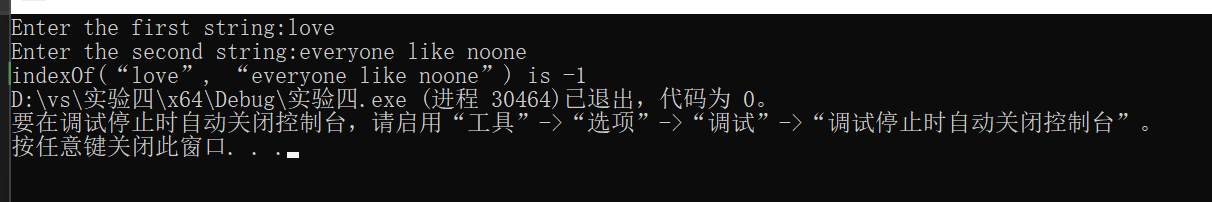
cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is " << indexOf(s1, s2);

return 0;

}

**Result：**

****

****

**第六题：**

**Code：**

#include<iostream>

using namespace std;

const int letter = 26;

const int n = 100;

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

{

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

{

counts[i] = 0;

}

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

{

counts[s[i] - 'a']++;

counts[s[i] - 'A']++;

}

}

int main()

{

char s[n];

int counts[letter];

cout << "Enter a string:";

cin.getline(s, n);

count(s, counts);

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

{

if (counts[i] != 0)

{

cout << char('a' + i) << ":" << counts[i] << "times" << endl;

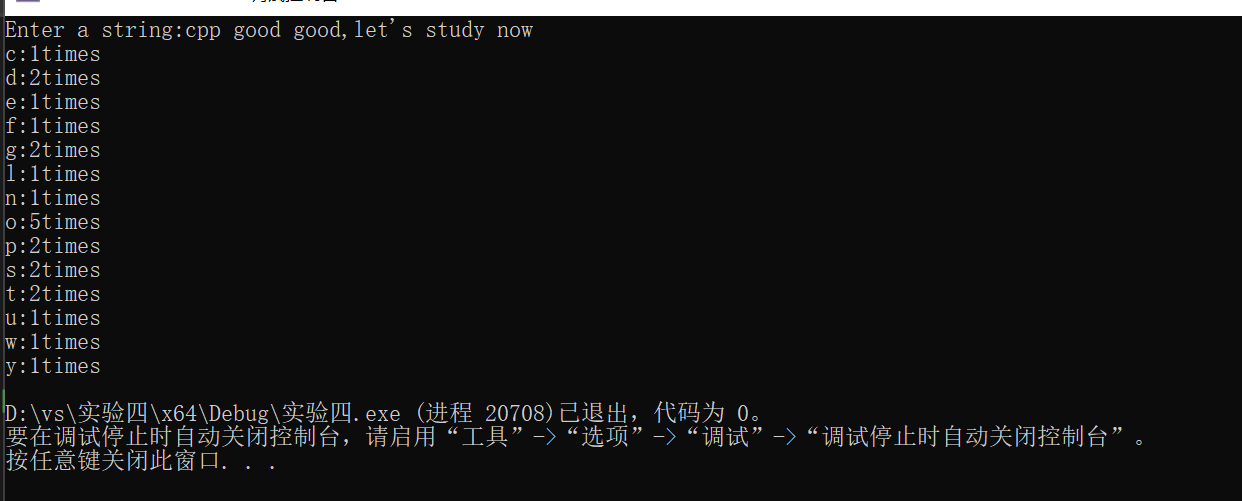
}

}

return 0;

}

**Result：**

****

1. **指针**

**第一题：**

**Code：**

#include<iostream>

using namespace std;

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

{

int len1 = strlen(s1), len2 = strlen(s2);

for (int i = 0; i <= len2 - len1; i++)//找第一个相同点

{

if (\*(s1+0) == \*(s2+i))//成功找到初始相同点

{

for (int j = 0; j < len1; j++)

{

if (\*(s1+j) != \*(s2+j+i))//发现不同

{

break;

}

if (j == len1 - 1)//全部核对

{

return i;

}

}

}

}

return -1;//第一个相同点未找到

}

int main()

{

const int maxn = 100;

char s1[maxn], s2[maxn];

cout << "Enter the first string:";

cin.getline(s1, maxn);

cout << "Enter the second string:";;

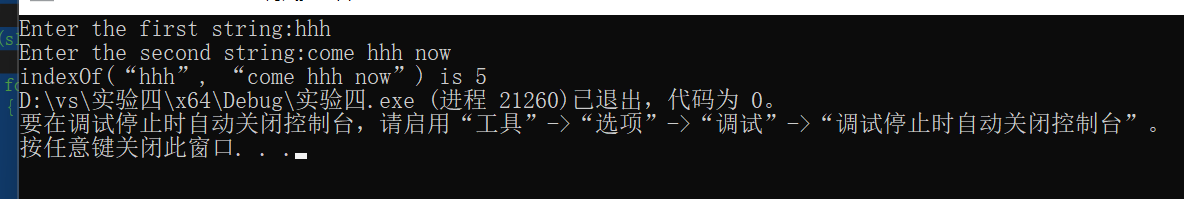
cin.getline(s2, maxn);

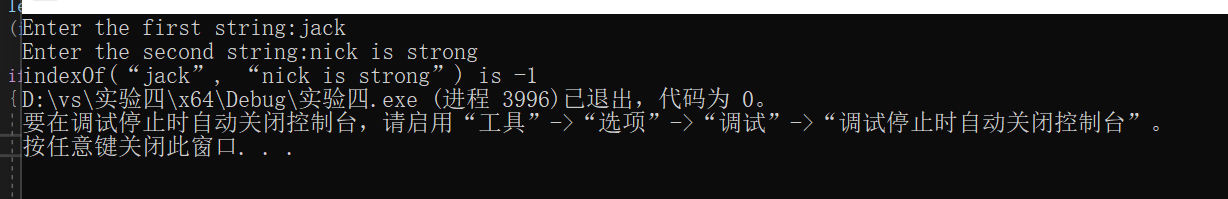
cout << "indexOf(“" << s1 << "”, “" << s2 << "”) is " << indexOf(s1, s2);

return 0;

}

**Result：**

****

****

**第二题：**

**Code：**

#include<iostream>

#include<cmath>

using namespace std;

const int maxn = 100;

int parseHex(const char\* const hexString)

{

int n=0,num[maxn],len=strlen(hexString);

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

{

switch (hexString[i])

{

case 'A':

num[i] = 10;

break;

case 'B':

num[i] = 11;

break;

case'C':

num[i] = 12;

break;

case'D':

num[i] = 13;

break;

case'E':

num[i] = 14;

break;

case'F':

num[i] = 15;

break;

default:

num[i] = int(hexString[i] - '0');

break;

}

}

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

{

n = n + num[i] \* pow(16, len - i - 1);

}

return n;

}

int main()

{

char hex[maxn];

cout << "enter hex:";

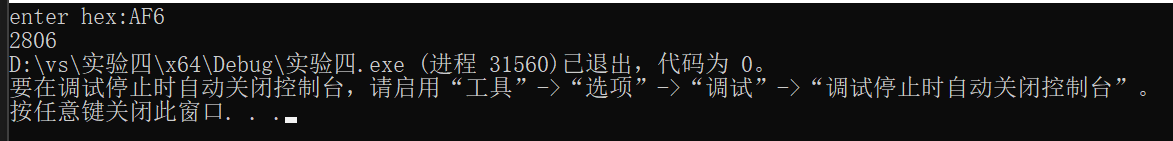
cin.getline(hex, maxn);

cout << parseHex(hex);

return 0;

}

**Result：**

****

**第三题：**

**Code：**

#include<iostream>

using namespace std;

void sort(int list[], int listsize)

{

bool changed = true;

do

{

changed = false;

for (int j = 0; j < listsize - 1; j++)

{

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

{

int t = list[j];

list[j] = list[j + 1];

list[j + 1] = t;

changed = true;

}

}

} while (changed);

}

int main()

{

int n;

int\* a = new int[5];

cin >> n;

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

{

cin >> a[i];

}

sort(a, n);

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

{

cout << \*(a + i) << " ";

}

delete[]a;

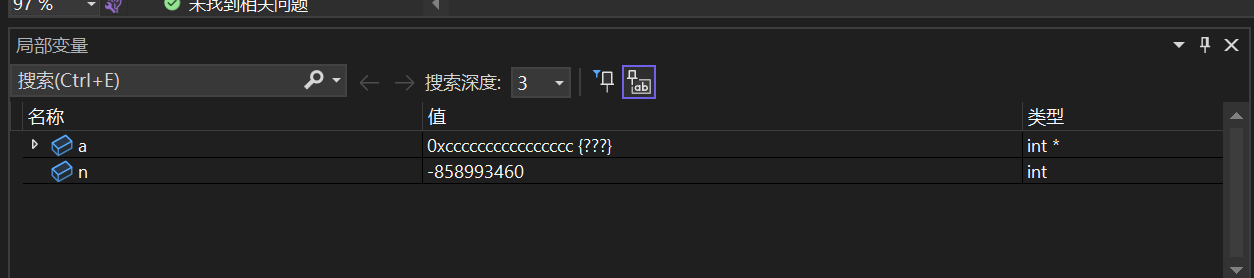
return 0;

}

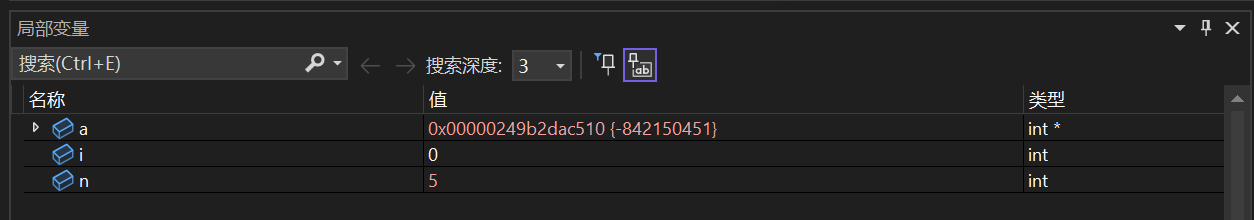
**Result：**

分别于分配内存，输入数组，释放内存前后四处建立断点，观察如下：

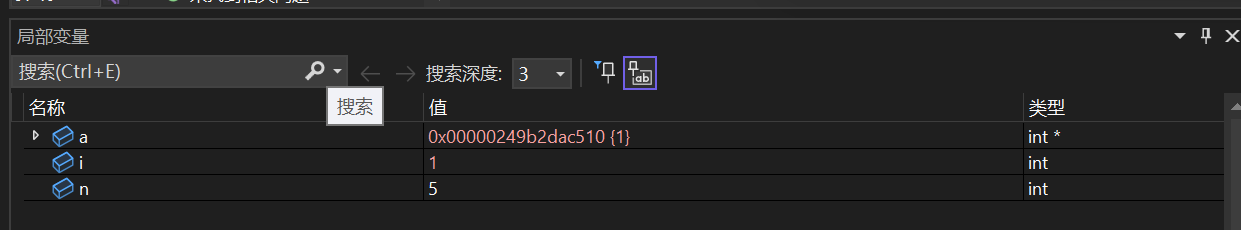
**分配内存前：**



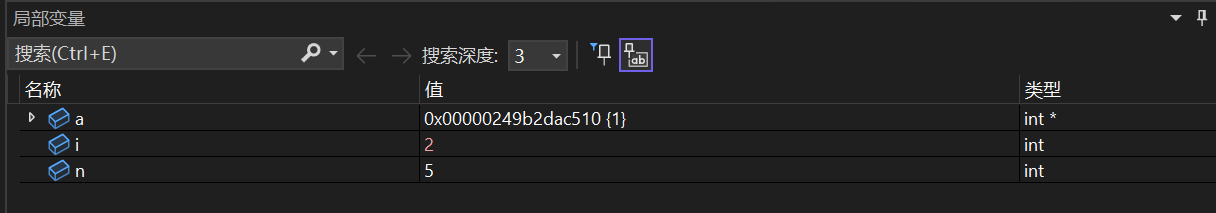
**分配内存后：**

****

**输入数组第一个元素后：**

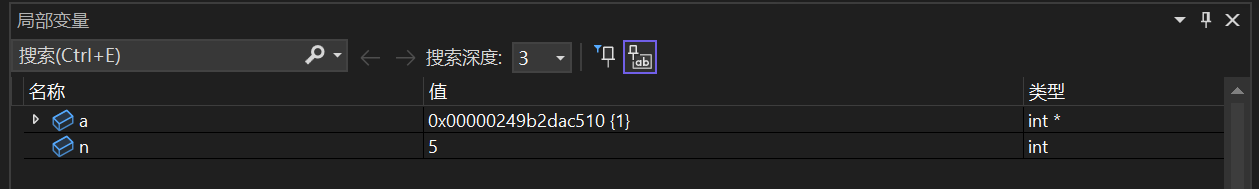
****

**输入数组第二个元素后：**

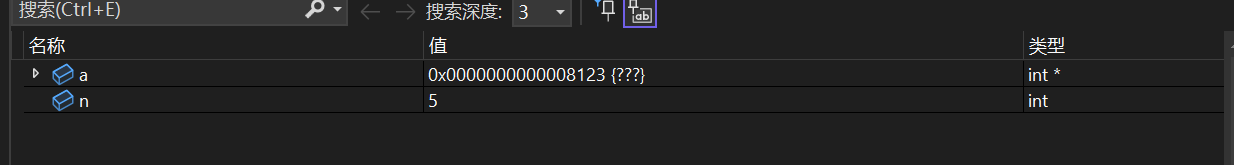
****

**剩下三个数组元素输入同上**

**释放内存前：**

****

**释放内存后：**

****

1. **遇到的问题和解决方法**

数组第六题在程序流程上遇到一定困难，既是因为题目中下标未能完全理解，也有长时间写代码后疲劳的原因，但在重新细读分解题目需求后，缕清了需要判断的东西与前后逻辑关系，根据分析得到的逻辑关系，成功解决问题。

1. **体会**

编程虽然只是简单的敲键盘，但背后的逻辑与抽象思考同样是很累的，不过这也使得每完成一个程序，就有强烈的充实感，激励自己不断编程下去，同时也通过这一实验更好理解了指针与数组，对二者的运用更加娴熟有把握。