s

**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 8209220610

姓名 王佳俊

**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）。

Sample in 1 2 3 2 1 6 3 4 5 2

Sample out 1 2 3 6 4 5

2.起泡排序：

编写测试程序，读入一个含有 10 个双精度数字的数组，调用bubble\_sort函数并显示排列后的数字。

参见 ：<https://oi-wiki.org/basic/bubble-sort/>

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。

假定数组大小不超过 80。

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

Sample in:

Enter list1: 5 1 5 16 61 111

Enter list1: 4 2 4 5 6

Sample out:

The merged list is 1 2 4 5 5 6 16 61 111

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[])

Sample in:

welcome

We welcome you!

Sample out:

indexOf(“welcome”, “We welcome you!”) is 3

Sample in :

Welcome

We invite you!

Sample out:

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。

Sample in:

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>

#include <algorithm>

using namespace std;

#pragma warning(disable:4996)

int main() {

int a[10];

printf("Enter ten numbers: \n");

for (int i = 0; i < 10; i++)

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

sort(a, a + 10);

auto k = unique(a, a + 10) - a;

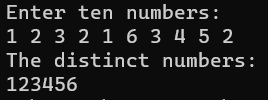
puts("The distinct numbers:");

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

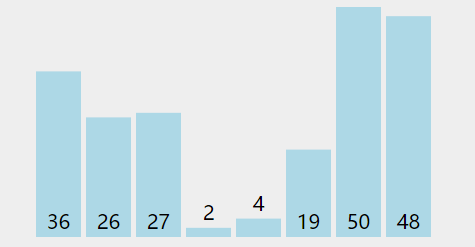
printf("%d", a[i]);

return 0;

}

**Result：**

**第二题：**



**Code：**#include <iostream>

#include <algorithm>

using namespace std;

#pragma warning(disable:4996)

void bubble(double \* a, int n)

{

bool flag = false;

while (!flag) {

flag = true;

for (int i = 0; i != n - 1; i++) {

if (a[i] > a[i + 1]) {

flag = false;

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

}

}

}

}

int main() {

double a[10];

puts("enter ten numbers");

for (int i = 0; i < 10; i++)

{

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

}

bubble(a, 10);

for (auto q : a)

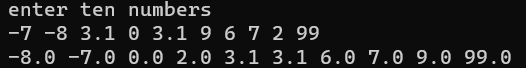
{

printf("%.1lf ", q);

}

return 0;

}

**Result：**

**第三题：**

**Code：**

#include <iostream>

using namespace std;

#pragma warning(disable:4996)

int main() {

bool a[101];

for (auto& q : a) q = 1;//after first guy

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

{

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

{

a[k] ^= 1;

}

}

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

{

if (a[i] == true)

{

printf("%d ", i);

}

}

return 0;

}

**Result：**

**第四题：**

**Code：**

#include <iostream>

using namespace std;

#pragma warning(disable:4996)

void shell\_sort(int array[], int length) {

int h = 1;

while (h < length / 3) {

h = 3 \* h + 1;

}

while (h >= 1) {

for (int i = h; i < length; i++) {

for (int j = i; j >= h && array[j] < array[j - h]; j -= h) {

std::swap(array[j], array[j - h]);

}

}

h = h / 3;

}

}

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

{

const int INF = 1 << 30;

for (int i = 1; i <= size1; i++)

{

list3[i] = list1[i-1];

}

for (int i = 1; i <= size2; i++)

{

list3[size1 + i] = list2[i-1];

}

shell\_sort(list3, size1 + size2);

for (int i = 1; i <= size1 + size2; i++)

printf("%d ", list3[i]);

}

int main() {

int a[81], b[81], c[165], asize, bsize;

printf("enter the size of a array of nondecreasing numbers,then enter the numbers one by one\n");

scanf("%d", &asize);

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

{

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

}

printf("once again\n");

scanf("%d", &bsize);

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

{

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

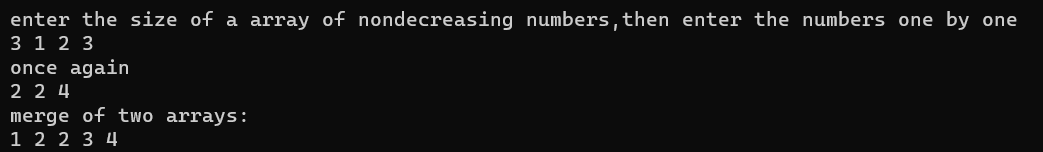
}

printf("merge of two arrays:\n");

merge(a, asize, b, bsize,c);

return 0;

}

**Result：**

**第五题：**

**Code：**

#include <iostream>

#include <string>

using namespace std;

#pragma warning(disable:4996)

int a, b;

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

{

if (a > b) return -1;

else {

for (int i = 0; i <= b - a; i++)

{

for (int q = 0; q < a; q++)

{

if (q == a - 1 && s1[q] == s2[i + q]) {

return i;

}

if (s1[q] != s2[i + q])

{

break;

}

}

}

}

return -1;

}

int main() {

string s1, s2;

getline(cin,s1);

getline(cin, s2);

a = s1.size(), b = s2.size();

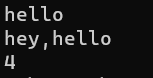
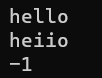
const char \*cs1 = s1.c\_str();

const char \*cs2 = s2.c\_str();

printf("%d", indexof(cs1, cs2));

return 0;

}

**Result： **

**第六题：**

**Code：**

#include <iostream>

#include <string>

using namespace std;

#pragma warning(disable:4996)

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

{

int i = 0;

char a = 'a';

char p = s[i];

while (p != '\0')

{

p = tolower(p);

if (isalpha(p))

{

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

{

if (p == a + i)

{

counts[i]++;

break;

}

}

}

p = s[++i];

}

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

{

if (counts[i] != 0)

{

printf("%c: %dtimes\n", a + i, counts[i]);

}

}

}

int main() {

int counts[26];

for (auto& q : counts) q = 0;

string s;

puts("enter a sentence");

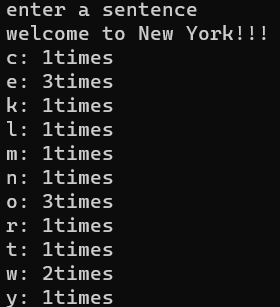
getline(cin, s);

const char\* cs = s.c\_str();

count(cs, counts);

return 0;

}

**Result：**

1. **指针**

**第一题：**

**Code：**

#include <iostream>

#include <string>

using namespace std;

#pragma warning(disable:4996)

int a, b;

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

{

if (a > b) return -1;

else {

for (int i = 0; i <= b - a; i++)

{

for (int q = 0; q < a; q++)

{

if (q == a - 1 && s1[q] == s2[i + q]) {

return i;

}

if (s1[q] != s2[i + q])

{

break;

}

}

}

}

return -1;

}

int main() {

string s1, s2;

getline(cin,s1);

getline(cin, s2);

a = s1.size(), b = s2.size();

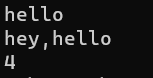
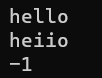
const char \*cs1 = s1.c\_str();

const char \*cs2 = s2.c\_str();

printf("%d", indexof(cs1, cs2));

return 0;

}

**Result： **

**第二题：**

**Code：**

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

#pragma warning(disable:4996)

int len;

int parseHex(const char\* const hexString)

{

int cnt = 0;

for (int i = len - 1; i >= 0; i--)

{

if (isalpha(hexString[i]))

{

cnt += pow(16,len - 1 - i) \* (hexString[i] - 'A' + 10);

}

else

cnt += pow(16, -i + len - 1) \* (hexString[i] - '0');

}

return cnt;

}

int main() {

puts("enter one hex number");

string s;

getline(cin, s);

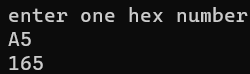
len = s.size();

const char\* cs = s.c\_str();

printf("%d", parseHex(cs));

return 0;

}

**Result：**

**第三题：**

**Code：**

#include <iostream>

#include <algorithm>

using namespace std;

#pragma warning(disable:4996)

void mysort(int\* a,int n)

{

for (int i = 0; i != n; i++) {

int ith = i;

for (int j = i + 1; j != n; j++) {

if (a[ith] > a[j]) {

ith = j;

}

}

swap(a[i], a[ith]);

}

}

int main() {

int n;

puts("enter the number of your sorted numbers");

scanf("%d",&n);

int\* a = new int[n];

puts("enter sorted numbers");

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

{

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

}

mysort(a, n);

int\* i = &a[0];

while (n--)

{

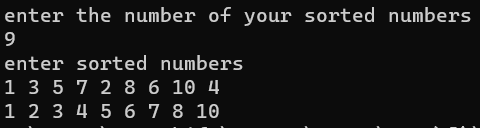
printf("%d ", \*i++);

}

delete [] a;

return 0;

}

**Result：**

1. **遇到的问题和解决方法**
2. 数组在传入函数参数时在函数体内将退化为指针，此时在函数体内使用sizeof将得到指针所占字节，故建议将数组长度一并设置为函数参数传入函数。
3. 字符串可以“任意长”，拷贝效率低且占空间，必要时考虑引用而非拷贝。
4. 对string型使用sizeof得不到string变量所占字节而是一个固定（相对固定的编辑器而言）的值，要使用 size函数。
5. **体会**
6. 对于数组out of range的问题，良好的编辑代码风格对减少错误的发生有一定帮助，较一致的比较取等有助于排查错误。
7. 指针复杂，难以驾驭，部分情况使用STL更方便算法实现。
8. 在使用指针时不要一味大胆猜测，可以逐步使用注释注释部分代码，步步排查疏漏同时加深对指针调用的理解。
9. 设置不使用指针的函数对拍观察指针与传统方式的不同于相同。