**实验三 函数**

**一．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；

6、掌握递归函数的定义方法。

**二．实验内容与要求**

（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。2. 编写程序满足：声明一个函数，判断一个整数是否为素数，使用如下函数头：

bool is\_prime(int num) ,如果 num 是素数函数返回 true，否则返回 false；

利用函数 is\_prime 找出前 200 个素数，并按每行 10 个输出：

2 3 5 7 11 13 17 19 23 29

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（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

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。

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。

（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’<<pj;

cout<<&i<<’\t’<<\*&i<<’\t’<<&j<<’\t’<<\*&j;

}

(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) 相当于 a[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)

{

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 \*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）。

**三．算法分析、程序与结果**

**1.程序：**

#include <iostream>

using namespace std;

void f(char \*st,int i)

{

st[i]='\0';

cout<<st; // printf(“%s\n”,st);

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

}

int \*f()

{

int \*p=new int [4]{1,2,3,4};

return p;

}

int main()

{

{

//（1）

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'<<pj;

cout<<&i<<'\t'<<\*&i<<'\t'<<&j<<'\t'<<\*&j;

cout<<'\n';

}

{

//(2)

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

}

{

//(3)

char st[]="abcd";

f(st,4);

cout<<'\n';

}

{

//(4)

int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

}

**2.（1）程序：**

#include <iostream>

#include <string>

#include <cstring>

#include <cmath>

using namespace std;

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

{

string str1=s1;

string str2=s2;

return str2.find(s1);

}

int main()

{

char s1[100],s2[100];

cout<<"Enter the first string:";

cin>>s1;

cout<<"Enter the second string:";

cin>>s2;

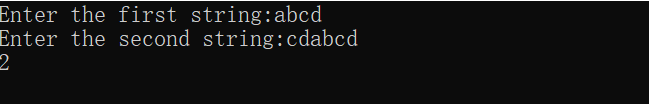
cout<<indexOf(s1,s2)<<endl;

return 0;

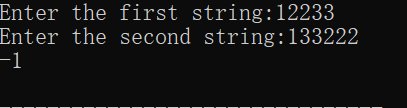
}

**输入输出：**

abcd cdabcd



12233 133222



**2.(2)程序：**

#include <iostream>

#include <cstring>

#include <cmath>

using namespace std;

int parseHex(const char \*const hexString)

{

int len=strlen(hexString);

int dec=0;

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

if (hexString[i]>='a'&&hexString[i]<='z') {

dec+=(hexString[i]-'a'+10)\*pow(16.0,len-1-i);

}

if (hexString[i]>='A'&&hexString[i]<='A') {

dec+=(hexString[i]-'A'+10)\*pow(16.0,len-1-i);

}

if (hexString[i]>='0'&&hexString[i]<='9') {

dec+=(hexString[i]-'0')\*pow(16.0,len-1-i);

}

}

return dec;

}

int main()

{

char hexString[100];

cin>>hexString;

cout<<parseHex(hexString);

return 0;

}

**输入输出：**

a5



12



**2.(3)程序：**

#include <iostream>

using namespace std;

void sort(int \*p,int n)

{

int t;

for (int i=1;i<n;i++) {

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

if (p[j]>p[i]) {

t=p[i];

p[i]=p[j];

p[j]=t;

}

}

}

}

int main()

{

int n;

cout<<"输入数组中元素个数";

cin>>n;

int \*p=new int [n];

cout<<"输入元素";

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

cin>>p[i];

}

sort(p,n);

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

cout<<p[i]<<' ';

}

delete p;

return 0;

}

**输入输出：**



**四．遇到的问题及解决方法**

合并数组时不会用限定的比较次数实现，通过上网查询解决；不会编写检验子串的程序，上网查到了解决方法；16进制字符串转化为10进制数时运算结果出错，多次调试解决了问题。