**计算机程序设计基础（C++)**

**实验报告**

专业班级： 软工2404

学 号： 82092428

姓 名： 龚浩天

**实验报告成绩：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验** | **实验一** | **实验二** | **实验三** | **实验四** | **实验五** | **总评** |
| **成绩** |  |  |  |  |  |  |

**批阅教师：**

**实验三 函数**

**一、实验目的**

本实验主要培养、训练学生对函数的理解，要求：

1. 掌握函数的定义、声明的方法；

2. 掌握函数的编写要求；

3. 掌握函数的调用方法；

4. 掌握函数参数的传递方法；

5. 掌握变量的作用域；

6. 掌握多文件编程方法。

**二、实验内容与要求**

1、输入自然数m和n，

（1）求他们的最大公约数（或称最大公因数）。

#include<iostream>

using namespace std;

//最大公因数函数

int max\_(int m, int n)

{

int t=0;

for (int i = 1;(i <= m || i <= n);i++)

{

if (m % i == 0 && n % i == 0)

{

t = i;

}

}

return t;

}

int main()

{

int a, b;

cout << "输入两个数，我会为你求出它们的最大公因数！" << endl;

cin >> a >> b;

cout << "最大公因数为：" << max\_(a, b) << endl;

return 0;

}

要求输入、输出在主函数中进行，求公约数由函数实现。

1. 在函数中求最大公约数与最小公倍数。（提示：使用引用参数）

#include<iostream>

using namespace std;

int big(0), small(0);

//函数体

void max\_min\_(int m, int n)

{

int&t = big;

for (int i = 1;(i <= m || i <= n);i++)

{

if (m % i == 0 && n % i == 0)

{

t = i;

}

}

for (int i = 1;!(i % m == 0 && i % n == 0);i++)

{

small= i;

}

}

int main()

{

int a, b;

cout << "输入两个数，我会为你求出它们的最大公因数和最小公倍数！" << endl;

cin >> a >> b;

max\_min\_(a, b);

cout << "最大公因数和最小公倍数分别为为：" <<big<<"与" <<small<< endl;

return 0;

}

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

#include<iostream>

using namespace std;

bool is\_prime(int num)

{

int i = 1;

do { i++; } while (!(num % i == 0));

if (i == num)

{

return 1;

}

else

{

return 0;

}

}

int main()

{

int a;

cout << "请输入一个数字，我会为你判断是否为素数！" << endl;

cin >> a;

if (is\_prime(a))

{

cout << a << "是素数！" << endl;

}

else {

cout << a << "不是素数！" << endl;

}

cout << "我会为你列出100以内的素数哦！" << endl;

for (int i(2),t(0);i <= 100/\*可以换成别的数\*/;i++)

{

if (is\_prime(i))

{

cout << i << '\t';

t++;

}

if (t % 10 == 0)

{

cout << endl;

}

}

return 0;

}

本实验员选择4完成！

3、编程实现摄氏温度到华氏温度的转换：

编写一个头文件，包含下面两个函数：

double celsius\_to\_fah(double cel)    //摄氏温度到华氏温度

double fahrenheit\_to\_cels(double fah) //华氏温度到摄氏温度

实现头文件，并编写测试程序，调用函数显示如下结果：

Celsius    Fahrenheit   |   Fahrenheit       Celsius

40.0       105.0        |   120.0            48.89

39.0       102.0        |   110.0            43.33

……       ……        |   ……             ……

31.0        87.8        |   30.0             -1.11

（测试程序为主模块，即main( )函数所在的CPP文件，头文件mytemperature.h只有函数声明；函数定义写在另一CPP文件mytemperature.cpp）

4、创建名为mytriangle.h的头文件，包括：

bool is\_valid(double side1,double side2,double side3)

double\_area(double side1,double side2, double side3)

面积=sqrt(s(s-side1)(s-side2)(s-side3))

其中s=(side1+side2+side3)/2

写测试程序：读取三角形三边长，如输入合法，计算面积，否则输出错误信息。

（测试程序为主模块，即main( )函数所在的CPP文件，头文件mytriangle.h只有函数声明；函数定义写在另一CPP文件mytriangle.cpp）

//mytriangle.h

#include<iostream>

using namespace std;

bool is\_valid(double side1, double side2, double side3)

{

int max\_ = (side1 > side2 ? side1 : side2)>side3? (side1 > side2 ? side1 : side2):side3;

return (max\_ < (side1 + side2 + side3 - max\_)) ? 1 : 0;

}

double area(double side1, double side2, double side3)

{

if (is\_valid(double side1, double side2, double side3))

{

double s = (side1 + side2 + side3) / 2;

return (sqrt(s(s - side1)(s - side2)(s - side3)))

}

else

{

cout << "(结果为0代表其实并不存在你所想的三角形哦...)";

return 0;

}

}

//mytriangle.cpp

#include<iostream>

#include"mytriangle.h"

using namespace std;

int main()

{

double a, b, c;

cout << "请输入三角形三边长，我使用海伦·秦九韶公式会为你计算其面积！" << endl;

cin >> a >> b >> c;

cout << area(a, b, c) << endl;

return 0;

}

**3与4选一个完成**

5、猴子吃桃：猴子第一天摘若干桃子，当即吃了一半，还不过瘾，又吃了一个。第二天又将剩下的桃子吃掉一半，又多吃一个，以后每天如此，到第10天，发现只剩最后一个桃子，问，第一天猴子共摘多少桃子（用递归实现）。

#include<iostream>

using namespace std;

int eat\_peach\_monkey(int day)

{

int num;

if (day == 1)

{

num = 1;

}

else

{

num = (eat\_peach\_monkey(day - 1) + 1) \* 2;

}

return num;

}

int main()

{

int a=(10);

//cin >> a;

cout << "一开始一共有" << eat\_peach\_monkey(a) <<"个桃子！" << endl;

return 0;

}

**三、实验思考题**

1. 本实验中函数中返回的值为什么与函数类型一致？

答：**return 语句中表达式的类型应与定义函数时指定的返回值类型一致**。 如果不一致，则以函数定义时的返回值类型为准，对 return 语句中表达式的类型自动进行转换，然后再将它返回给主调函数使用。 在调用函数时，如果需要从被调函数返回一个值供主调函数使用，那么返回值类型必须定义成非 void 型。 此时被调函数中必须包含 return 语句，而且 return 后面必须要有返回值，否则就是语法错误

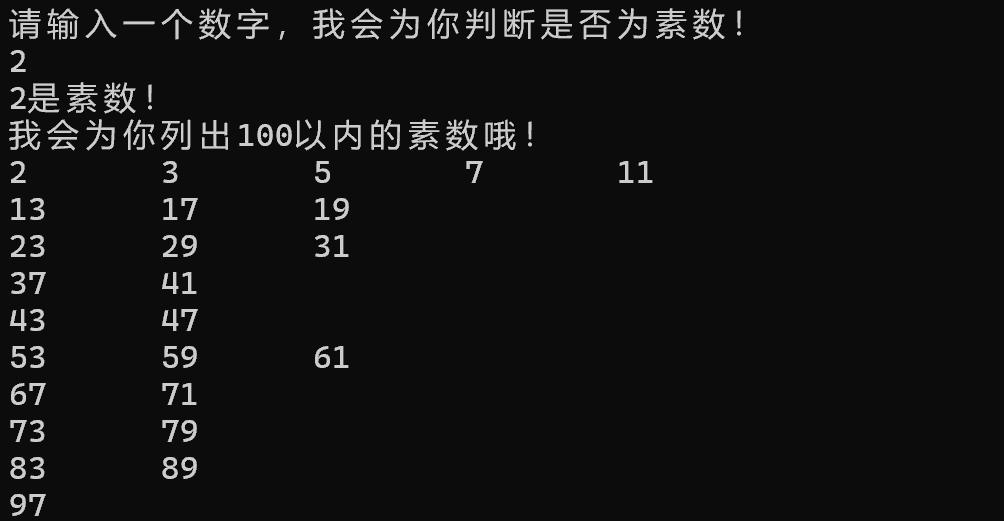
2. 本实验中主函数调用函数时采用的是何种传递方式？

答：值传递

**四、算法分析，程序结果**

**五、遇到的问题与解决方法**

**1，在实验2中，遇到10个一换行的输出结果与预期不一致：**

****

**解决方法：**

**经过**

**六、体会**

**在实验中，我真正理解了函数是如何被定义和声明的。明确了函数就像是一个小型的程序模块，它有特定的输入（参数）和输出（返回值或无返回值的操作结果）。学会了如何准确地定义函数的返回值类型、函数名以及参数列表，这让我能够根据不同的需求创建出具有特定功能的代码块。同时，理解了函数声明的重要性，它就像是在代码中提前给编译器一个 “预告”，让程序在调用函数之前就知道这个函数的存在，避免出现编译错误。**

**掌握函数的调用方法让我能够在程序的不同部分灵活地使用已定义的函数。通过传入合适的参数，能够得到函数的输出结果，实现了代码的复用和模块化。在调用函数时，需要注意参数的类型和数量要与函数定义相匹配，否则会出现错误。这也让我更加注重代码的严谨性和准确性。**

**认识到变量的作用域对于编写清晰、正确的代码至关重要。局部变量只能在其所在的函数内部使用，而全局变量则可以在整个程序中访问。合理地使用变量的作用域可以避免变量名冲突，提高代码的可维护性。在实验中，我学会了如何根据需要选择合适的变量作用域，以及如何避免不必要的变量冲突。**

**All in all，这次实验让我在编程能力和思维方式上都有了很大的提升。我将继续努力，不断深化对函数以及其他编程概念的理解和运用，为今后的编程学习和实践打下坚实的基础。**

**实验四 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；

6、掌握递归函数的定义方法。

【实验内容与步骤】

**（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

Enter ten numbers: 1 2 3 2 1 6 3 4 5 2

The distinct numbers are: 1 2 3 6 4 5

#include<iostream>

using namespace std;

int main()

{

int a[10]{ 0 }, b[10]{ 0 };

cout << "Enter ten numbers:";

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

{

cin >> a[i];

}

//检索原理为对比每一个数组中的元素，若不重复n++,重复不执行n++

//最后通过判断n的大小来间接判断是否为重复数字

int m = 0;//不重复数字的个数

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

{

int n = 0;//初始化判断计数变量

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

{

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

n++;

}

if (n == 1)

{

b[m] = a[i];//将不重复的数输入b数组，并使计数变量加一；

m++;

}

}

cout << "The distinct numbers are:";

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

{

cout << b[i] << " ";

}

return 0;

}

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。

#include<iostream>

using namespace std;

static void bubble\_sort(double a[10])

{

double b[1] = {0};

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

{

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

{

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

{

b[0] = a[i];

a[i] = a[j];

a[j] = b[0];

}

}

}

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

{

cout << a[i] << '\t';

}

}

int main()

{

double c[10]{};

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

{

cin >> c[i];

}

bubble\_sort(c);

return 0;

}

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。

#include<iostream>

using namespace std;

int main()

{

bool L[100]{0};

//第一个学生打开所有储物柜

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

{

L[i] = 1;

}

//第二个学生开始对储物柜的操作：

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

{

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

{

L[n]=!L[n];

}

}

//输出所有柜子的状态（1代表开，0代表关，每10个换一行）

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

{

cout << L[i-1] << " ";

if (i % 10 == 0)

{

cout << endl;

}

}

return 0;

}

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

Enter

Enter list1: 5 1 5 16 61 111

Enter list1: 4 2 4 5 6

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

#include<iostream>

using namespace std;

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

{

int a(0), b(0);

for (int i = 0;i < (size1 + size2);i++)//进行数组合并的循环操作：

{

if (list1[a] > list2[b])

{

list3[i] = list2[b];

if ((b + 1) >= size2)//判断b数组是否输入完毕

{

i++;

for (;i < (size1 + size2);i++)//若输入完毕，则剩下的a数组全盘输入！

{

list3[i] = list1[a];

a++;

}

}

else b++;

}

else

{

list3[i] = list1[a];

if ((a + 1) >= size1)//判断a数组是否输入完毕

{

i++;

for (;i < (size1 + size2);i++)//若输入完毕，则剩下的b数组全盘输入！

{

list3[i] = list2[b];

b++;

}

}

else a++;

}

}

for (int i = 0;i < (size1 + size2);i++)//输出合并后的数组：

{

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

if ((i + 1) % 10==0)

{

cout << endl;

}

}

}

int main()

{

int size1, size2;

int list1[80], list2[80], list3[160];

cout << "Enter list1:" << endl;

cin >> size1;

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

{

cin >> list1[i];

}

cout << "Enter list2:" << endl;

cin >> size2;

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

{

cin >> list2[i];

}

cout << "The merged list is:" << endl;

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

return 0;

}

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

Enter

Enter

Enter the first string: welcome

Enter the second string: We welcome you!

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

Enter

Enter the first string: welcome

Enter the second string: We invite you!

indexOf(“welcome”, “We invite you!”) is –1

#include<iostream>

#include<string>

using namespace std;

//先设计一个函数，读取s1中字符的个数

int charnumber(const char s1[])

{

int n(0);

for (int i = 0;s1[i] != '\0';i++)

{

n++;

}

return n;

}

//再设计函数读取子串

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

{

//空指针检验

if (s1 == nullptr || s2 == nullptr)

{

return -1;

}

int n(0);//重复字符数计数：

for (int i = 0;s2[i] != '\0';i++)

{

if (s1[n] == s2[i])

{

n++;

if (n == charnumber(s1))

{

return (i-n+1);

}

}

else

n = 0;

}

return -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 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

#include<iostream>

#include <cstring>

using namespace std;

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

{

for (int i = 0;s[i] != '\0'; i++)

{

if (s[i] >= 97 && s[i] <= 122)

{

counts[s[i] - 97]++;

}

if (s[i] >= 65 && s[i] <= 90)

{

counts[s[i] - 65]++;

}

}

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

{

if (counts[i] != 0)

{

cout << static\_cast <char>(97 + i) << " :" << counts[i] << " times" << endl;

}

}

}

int main()

{

int counts[26] = { 0 };

char s1[]{ 0 };

cout << "Enter a string:";

cin.getline(s1, 1000);//输入字符数组；

count(s1, counts);

return 0;

//在运行后可能报错

// 原因如下：

//C 风格的字符数组在使用时相对容易出现越界等错误

//而且编译器有时候可能无法提前检测出所有潜在的问题

//到运行时才暴露出栈被破坏这样的错误。

//（但是由于题目要求使用char[]作为形参...）

}

**（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;

}

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。

修改代码如下：

#include<iostream>

using namespace std;

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' << endl << pi << '\t' << pj << endl;

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

}

//运行结果：



(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

}

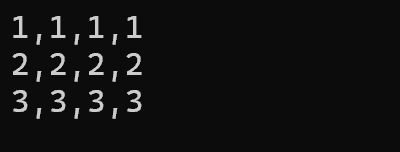
运行结果：

1,1,1,1

2,2,2,2

3,3,3,3

运行结果：



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\_\_\_\_abcdabcaba\_\_\_\_

(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()

{

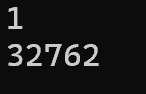
int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

}

运行结果：



并不是预期结果：

修改程序：

#include<iostream>

using namespace std;

int \*f()

{

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

return list;

}

void main()

{

int \*p=f();

cout<<p[0]<<endl;

cout<<p[1]<<endl;

delete[]p;

}

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);

#include<iostream>

#include<cstring>

using namespace std;

//先设计一个函数，读取s1中字符的个数

int charnumber(const char \*s1)

{

int n(0);

for (int i = 0;s1[i] != '\0';i++)

{

n++;

}

return n;

}

//再设计函数读取子串

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

{

//空指针检验

if (s1 == nullptr || s2 == nullptr)

{

return -1;

}

int n(0);//重复字符数计数：

for (int i = 0;s2[i] != '\0';i++)

{

if (s1[n] == s2[i])

{

n++;

if (n == charnumber(s1))

{

return (i-n+1);

}

}

else

n = 0;

}

return -1;

}

int main()

{

char s1[1000]{ 0 };

char s2[1000]{ 0 };

cout << "Enter the first string:";

cin.getline(s1, 1000);

cout << "Enter the second string: ";

cin.getline(s2, 1000);

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

#include<iostream>

#include<cmath>

using namespace std;

int parseHex(const char\* const hexString)

{

int x=strlen(hexString);//16进制位数

int sum=0;

for (int i =0;i<strlen(hexString);i++)

{

int t=0;//辅助存储器

if (hexString[i] >= 65 && hexString[i] <= 69)//大写字母转化

{

t = hexString[i] - 65 + 10;

}

else if (hexString[i] >= 97 && hexString[i] <= 102)//小写字母转化

{

t = hexString[i] - 97 + 10;

}

else if (hexString[i] >= 48 && hexString[i] <= 57)

{

t = (hexString[i]-48);

}

else

{

cout << "输入了错误的16进制字符" << endl;

t = 0;

break;

}

sum += (pow(16, x-i-1) \* t);

}

return sum;

}

int main()

{

char hexstring[1000]{ 0 };

cout << "请输入一个16进制的数，我将为你转化为十进制！" << endl;

cin >> hexstring;

cout << "转化结果是："<< parseHex(hexstring) <<endl;

return 0;

}

1. 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）。

**程序设计：**

**#include<iostream>**

**using namespace std;**

**void taketurn(int n,int\* ip)**

**{**

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

**{**

**for (int t = i+1;t < n;t++)**

**{**

**if (\*(ip + i) >= \*(ip + t))//比较两元素大小，将小的排在前面！**

**{**

**int s = 0;//交换中间变量；**

**s = \*(ip + t);**

**\*(ip + t) = \*(ip + i);**

**\*(ip + i) = s;**

**}**

**}**

**}**

**}**

**int main()**

**{**

**int n;**

**cout << "请输入数组存储元素的个数" << endl;**

**cin >> n;**

**int\* ip = new int[n];**

**cout << "请依次输入数组元素" << endl;**

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

**{**

**cin >> \*(ip + i);**

**}**

**cout << "输入完毕，可以动态调试指针查看输入的第n个数字(输入0可以结束调试)" << endl;**

**cout << endl;**

**int t = 0;//调试变量**

**cin >> t;**

**for (;t != 0;)**

**{**

**if (0 < t && t <= n)**

**{**

**cout << \*(ip + (t - 1)) << endl;**

**cout << "再次输入可以再次查询：(输入0可以结束调试)" << endl;**

**cout <<endl;**

**cin >> t;**

**}**

**else**

**{**

**cout << "查询的值超出范围，请重试" << endl;**

**cin >> t;**

**}**

**}**

**cout << endl;**

**cout << "指针调试完毕，正在为您从小到大排序数组元素！" << endl;**

**cout << "输出数组：" << endl;**

**cout << endl;**

**taketurn(n, ip);//调用函数进行排序**

**//输出排列完毕的数组**

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

**{**

**cout << \*(ip + i) << '\t';**

**if ((i+1) % 5 == 0)**

**{**

**cout << endl;**

**//五个数据一换行，保持美观；**

**}**

**}**

**delete[]ip;**

**return 0;**

**}**

【完成实验报告】

**实验报告只要求写程序设计部分**

**三、算法分析，程序结果**

**四、遇到的问题与解决方法**

**1、对 （一）实验3 中的储物柜游戏的代码优化：**

**问题：是否存在数学模型，进而可以从看见本质问题后优化代码？**

**解决方法：经过分析后得到：**

**根据数学分析，本题目题意可知**

**非完全平方数的因数是成对存在的，因此经过偶数次操作，**

**储物柜门状态必然和第一位同学操作完一致；**

**完全平方数的因数总是奇数个，因此处于完全平方数的储物柜**

**一定只受到奇数次操作！即状态与第一位同学操作完相反！**

**由此可以（-面向答案编程-划掉）简化代码：**

**（注意s2是从L2开始操作的，故要+1）**

**int main ()**

**{**

**bool L[100]{1}**

**for (int i=1;i<=10;i++)**

**{**

**L[(i\*i)]=!L[(i\*i)]**

**}**

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

**{**

**cout << L[i - 1] << " ";**

**if (i % 10 == 0)**

**{**

**cout << endl;**

**}**

**}**

**return 0;**

**}**

**2、对实验（一）4中题目问题的遇见与解决：**

**在实验过程中，易出现的错误如下所示：**

**cout<<** **void merge(const int list1[], int size1, const int list2[], int size2, int list3[])<<endl;**

**由于在定义函数时，merge函数的类型是void，返回值为空，故不能通过cout来输出！这是对定义时函数类型的不理解；**

**解决方法：**

**直接修改语句：**

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

**3、对实验（一）5遇见的问题与解决：**

**实验员在一开始在 main 函数中使用 cin >> s1 和 cin >> s2 来读取用户输入的字符串时遇到问题：cin >> 操作符读取字符串时，它会以空白字符（空格、制表符、换行符等）作为分隔符，当遇到第一个空白字符时就停止读取。例如，如果用户输入 "Hello World" 这样包含空格的字符串，实际 s1 只会获取到 "Hello" 部分，而 "World" 及后面的内容会被丢弃，导致后续对完整字符串进行处理的逻辑可能出现错误，比如在调用 indexOf 函数时，传入的并不是期望的完整输入内容。**

**解决方法：**

**通过查阅资料**

**运用std::getline(cin, s1)和std::getline(cin, s2)代替cin即可解决：**

**（添加头文件#include<string>）**

**五、体会**

这次实验，是理论与实践深度碰撞，数组、指针等知识交织成网，勾勒程序底层运作轮廓，在实验之中的收获沉淀为编程路上宝贵财富，激励我继续深挖技术，用代码编织更精妙程序世界。