实验三 数组&指针&字符串

作者：李文骏　　学号：2020213969　　日期：2021-11-18

1.实验目的要求

(1)　掌握一维数组和二维数组的定义、初始化赋值、元素引用的方法。

(2)　初步了解数组名与地址的关系。

(3)　掌握字符数组和字符串函数的使用。

(4)　掌握与数组有关的算法（特别是排序和查找算法）。

(5)　掌握指针的概念，学会定义和使用指针变量。

(6)　学会使用数组指针和指向数组的指针变量。

(7)　学会使用字符串指针和指向字符串的指针变量。

(8)　了解指向指针的指针的概念以及其使用方法。

(9)　掌握指针、引用、数组做函数参数的传递机制。

2.实验设备与环境

硬件环境：

- Huawei Matebook 14锐龙版笔记本一台
- PU为AMD Ryzen 7 4800U with Radeon Graphics 1.80 GHz，内存为16GB，硬盘大小为463GB

软件环境：

- 操作系统：Windows10家庭中文版
- 开发工具：Visual Studio 2019

3.实验内容

（1）阅读下列程序，写出其运行结果，并指出其功能

<1>

```
#include <iostream>
using namespace std;

const int N = 10;
int  main()
{   int a[N];
    int s,i,j,t;
    //1如下这段代码功能？
    for(i = 0; i<N; i++){
        if (i%2)  a[i] = i*i ;
```

```
        else a[i] = 100-(i/2)*(i/2);
    }
   //2如下这段代码功能？
    for (i=0; i<N; i++ )
        cout<<a[i]<<" ";
    cout<<endl ;
//3如下这段代码功能？
    s = 0 ;
    for( i=0; i<N; i++ )
        s = s + a[i];
    cout<<"sum of a = "<<s;
    cout<<endl;
//4如下这段代码功能？
 for(i=0; i<N; i++){
        t = a[i] ;
        j = i-1 ;
        while(j>=0&&t>a[j]){
            a[j+1] = a[j] ;
            j-- ;
        }
        a[j+1] = t ;
    }
//5如下这段代码功能？
    for(i=0; i<N; i++)
        cout<<a[i] <<" ";
        cout<<endl ;
    return 0;
}
```

分别描述上面代码中各个部分的功能，并给出输出结果。

解答：

1.给数组a中的每个元素赋值，下标为奇数的元素赋值为其下标的平方，下标为偶数的元素赋值为100减去偶数除以2的平方

2.打印输出数组a中的每个元素，并以空格作为分隔，打印完所有数之后进行一次换行

输出如下：`100 1 99 9 96 25 91 49 84 81`

3.求出数组a所有元素的和

输出：`sum of a = 635`

4.将数组a按照从大到小的顺序逆序排列

5.打印数组a中的元素，每个元素用空格进行分隔，同时输出每个元素的都进行一次换行

输出如下：

```
100
99
96
91
84
81
49
25
9
1
```

<2>

```cpp
#include <iostream>
#include <string>
using namespace std;

int  main()
{   int i=0, base, n, j, num[30] ;
    cin>>n>>base ;
    do{ i++;
        num[i] = n%base ;
        n = n/base ;
    }while(n!=0);

    for(j=i; j>=1; j--){
        switch(num[j]){
            case 15:cout<<'F';break;
            case 14:cout<<'E';break;
            case 13:cout<<'D';break;
            case 12:cout<<'C';break;
            case 11:cout<<'B';break;
            case 10:cout<<'A';break;
            default: cout<<num[j];
        }
    }
    return 0;
}
```

- 如果输入15，2，输出什么?

预期输出：

`1111`

实际输出：

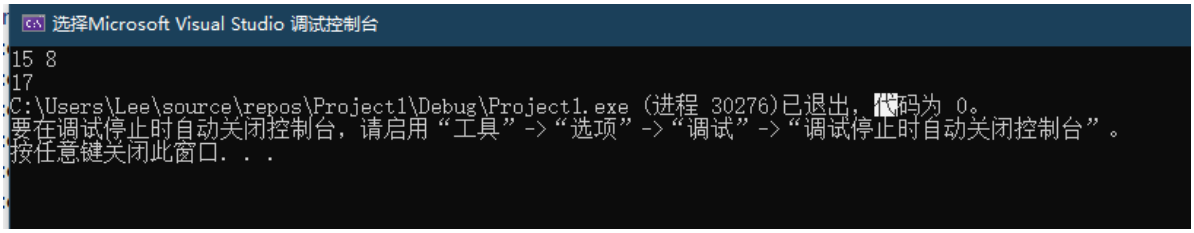
```
15 2
1111
C:\Users\Lee\source\repos\Project1\Debug\Project1.exe (进程 5404)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

- 如果输入15，8，输出什么?

预期输出：

17

实际输出：

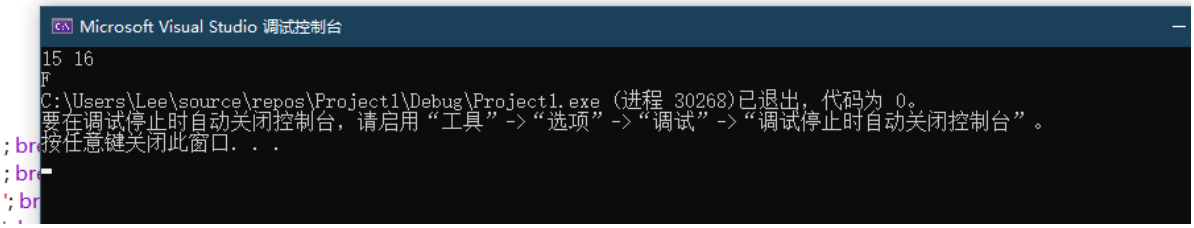


- 如果输入15，16，输出是什么?

预期输出：

F

实际输出：



总结如上代码的功能是什么?

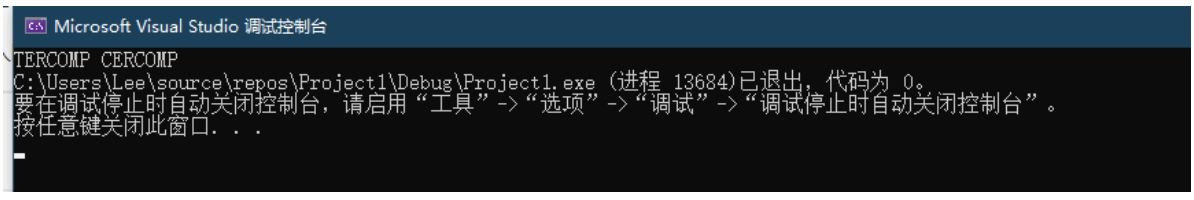
答：代码将输入的十进制数n转换成所输入的base进制的数，并输出

<3>

```
#include <iostream>
#include <string.h>
using namespace std;

int main(){
    char a[20] = "TER", b[20] = "COMP";
    int i = 0 ;
    strcat(a,b);
    while (a[i++]!='\0')
        b[i] = a[i] ;
    cout<<a <<" "<<b;
}
```

程序输出是什么，分析原因?

程序输出：



解答：strcat(m,n)函数的作用是将字符串m和字符串n连接之后，得到一个组合的字符串并赋值给m，故a的输出结果为**TERCOMP**

`while (a[i++]!='\0')`表达的是意思是先判断`a[i]!='\0'`，然后再进行`i++`的操作，最后往下进行循环体的语句，因此第一次循环时`i`的值为1，所以只从字符数组a的第二个元素开始复制给字符数组b，因此b的输出结果为**CERCOMP**

<4>

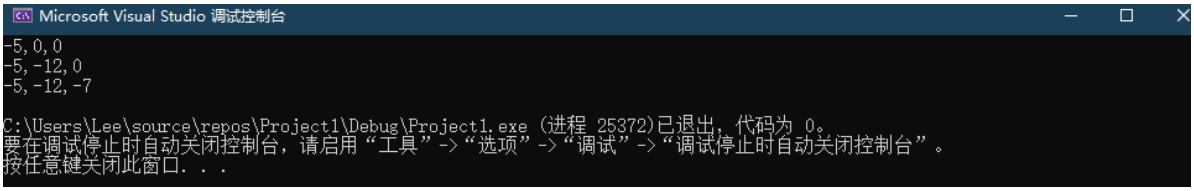
```
#include <iostream>
using namespace std;

void sub(int x, int y, int *z)
{   *z = y - x ;
}

int  main()
{   int a=0,b=0,c=0;
    sub(10, 5, &a);
    cout<<a<<','<<b<<','<<c<<endl;
    sub(7, a, &b);
    cout<<a<<','<<b<<','<<c<<endl;
    sub(a, b, &c);
    cout<<a<<','<<b<<','<<c<<endl;
    return 0;
}
```

- **Sub函数里第三个形参的作用是什么?**

解答：接收调用sub函数的传递的第三个实参的地址，使得sub函数调用能直接改变相应的值

- **主函数的输出结果是什么，分析下原因?**

解答：主函数输出结果如下：



原因分析：

- **第一次调用sub函数时**，第三个实参为变量a的地址，因此变量a的值发生了改变（变成了5-10的结果-5），其余变量，由于传入的不是变量地址，sub函数对它们的操作不能改变其变量本身的属性，因此输出`-5,0,0`
- **第二次调用sub函数时**，第三个实参为变量b的地址，因此变量b的值发生了改变（变成了a-7的结果，由于第一次调用sub函数使得变量a的初始值不再是0，而是第一次函数调用后的结果，即a此时的初始值为-5，结果也就为-5-7的结果，为-12），其余变量，由于传入的不是变量地址，sub函数对它们的操作不能改变其变量本身的属性，因此输出`-5,-12,0`
- **第三次调用sub函数时**，第三个实参为c的地址，因此变量c的值发生了改变（变成了第一次和第二次sub函数操作后，a和b变量的初始值分别为-12和-5，则第三次调用sub函数后，c的值为-12-（-5），即c的值为-7），其余变量，由于传入的不是变量地址，sub函数对它们的操作不能改变其变量本身的属性，因此输出`-5,-12,-7`

<5>

```cpp
#include <iostream>
#include <string.h>
using namespace std;

int main()
{   int stre(char[ ]) ;
    char str[10], *p = str ;
    gets(p);
    cout<<stre(p)<<endl;
    return 0;
}

int stre(char str[ ])
{   int num = 0 ;
    while(*(str + num)!='\0')  num++ ;
    return num;
}
```

- **Stre函数的功能是什么?**

计算字符数组中有多少个元素，并返回计算结果

- **如果你输入超过9个字符，发生了什么，结果会怎样？从数组越界的角度分析一下。**

如果输入超过9个字符时，由于字符串存储最后一位默认为`'\0'`，即若你输入10个字符，你的字符串的长度是11，超过了str数组的长度声明，因此会发生越界。

（2）编写程序实现下列问题的求解

<1> 编程产生下列数组，并输出。（至少选作两题，从一维数组和二维数组各选一题）

一维数组

1） (1 4 9 16 25 36 49 64 81 100)

2） (1 3 6 10 15 21 28 36 45 55)

3） (1 2 3 5 8 13 21 34 55 89)

二维数组

1）
$$\begin{pmatrix} 1&2&3&4&5&6\\ 2&3&4&5&6&0\\ 3&4&5&6&0&1\\ 4&5&6&0&1&2\\ 5&6&0&1&2&3\\ 6&0&1&2&3&4 \end{pmatrix}$$

2）
$$\begin{pmatrix} 1&2&3&\cdots&n-1&0\\ 2&3&4&\cdots&0&1\\ 3&4&5&\cdots&1&2\\ &&\cdots\cdots\cdots\cdots&&\\ n-1&0&1&\cdots&n-3&n-2\\ 0&1&2&\cdots&n-2&n-1 \end{pmatrix}$$

3）
$$\begin{pmatrix} 1&2&3&4&5&6\\ 2&1&2&3&4&5\\ 3&2&1&2&3&4\\ 4&3&2&1&2&3\\ 5&4&3&2&1&2\\ 6&5&4&3&2&1 \end{pmatrix}$$

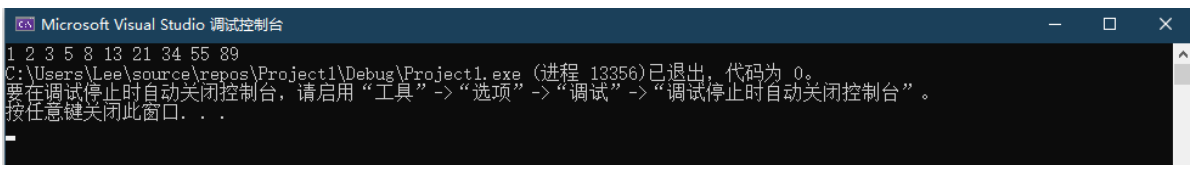
4）
$$\begin{pmatrix} A&B&C&D&E&F\\ B&C&E&H&L&Q\\ C&D&F&I&M&R\\ D&E&G&J&N&S\\ E&F&H&K&O&T \end{pmatrix}$$

一维数组中，选择3）：

代码如下：

```cpp
#include <iostream>
using namespace std;
int main() {
    int a[10];
    a[0] = 1;
    a[1] = 2;
    for (int i = 2; i < 10; i++) {
        a[i] = a[i - 1] + a[i - 2];
    }
    for (int i = 0; i < 10; i++) {
        cout << a[i] << " ";
    }
    return 0;
}
```

输出结果如下：

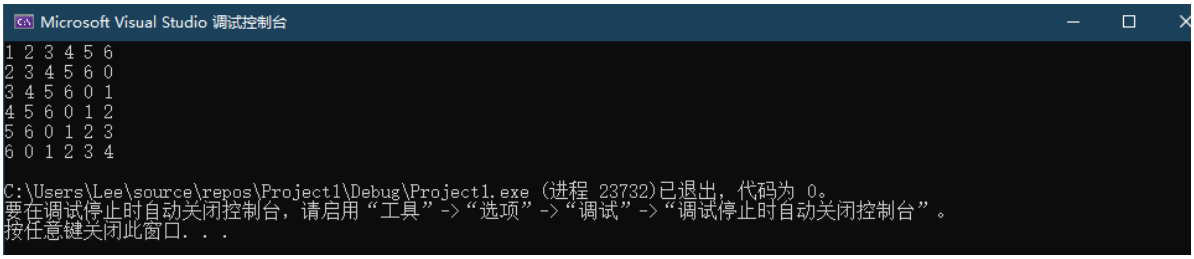


二维数组中，选择1）：

代码如下：

```cpp
#include <iostream>
using namespace std;
int main() {
    char a[6][6];
    string s = "1234560";
    int p = 0;
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 6; j++) {
            a[i][j] = s[p];
            p++;
            if (p == 7) {
                p = 0;
            }
        }
        p = i + 1;
    }
    //打印输出
    for (int i = 0; i < 6; i++) {
        for (int j = 0; j < 6; j++) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

输出结果如下：



<2>- <10> 至少选做5题

<2> 用**函数**实现将一个以字符串形式表示的十六进制数转换为一个十进制整数。例如，输入”A2”转换为162。

<3> 编写出判断一个整数是否为素数的**函数**，并求出在2000以内的有十个以上的所有连续的非素数组。

<4>将一个3*3的矩阵转置，用一个**函数**实现。在主函数中输入以下矩阵元素：{2，4，6，8，10，12，14，16，18}。将数组名作为函数参数。函数调用后在主函数中输出已转置的矩阵。

<5>自己写一个strcmp函数，实现两个字符串的比较。当s1<s2时，返回-1；当s1=s2时，返回值= 0；当s1>s2时，返回正数。两个字符串s1,s2由main函数输入，strcmp**函数**的返回值在main函数中输出。

<6> 判断一个二维数组是否有“鞍点”，即某位置上的元素在该行上最大，在该列上最小。如有，输出其行列号和值；若无，给出提示。

<7> 编程求一组整数的最大公因子和最小公倍数，讲任务分解为多个函数实现。

<8> 对于一个有序的数组，输入一个数，用折半查找法在数组中查找，如在数组中，则输出元素在数组中的位置；如不在，则输出-1。用函数实现。

<9> 用筛选法求出2到1000之间的所有素数，用函数实现。数组依次存储1~1000数据，函数运行结束后，该数组中非素数位置变为0，非零的素组元素即为素数。

注意： 筛选法求出2～N间的所有素数的方法是：首先将这些数全部放入一个数组中，然后重复下面的操作直到数组为空为止：

A）.找出其中的最小数K，则K一定是一个素数，因此可输出。

B）.从数组中删除K及其所有倍数。

<10> 在某系的成绩登记册中，每个班最多有40个学生，每份成绩表中的成绩信息包括：学号（9位字符），姓名（8位字符），成绩（百分制），备注（20位字符）。设计程序以处理一个班级的成绩信息，包括输入、输出、查询（给定分数以上或以下的学生信息）、按分数排序等。

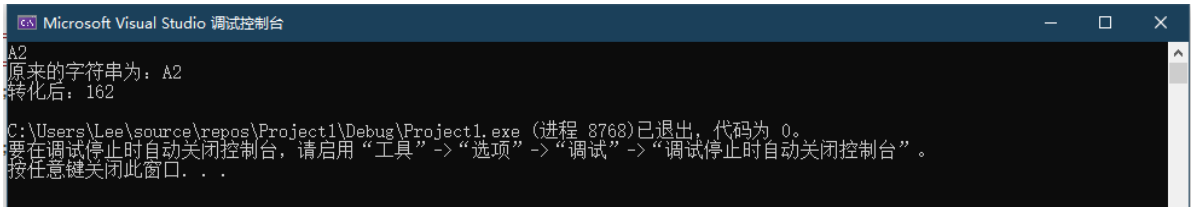
<2> 用**函数**实现将一个以字符串形式表示的十六进制数转换为一个十进制整数。例如，输入”A2”转换为162。

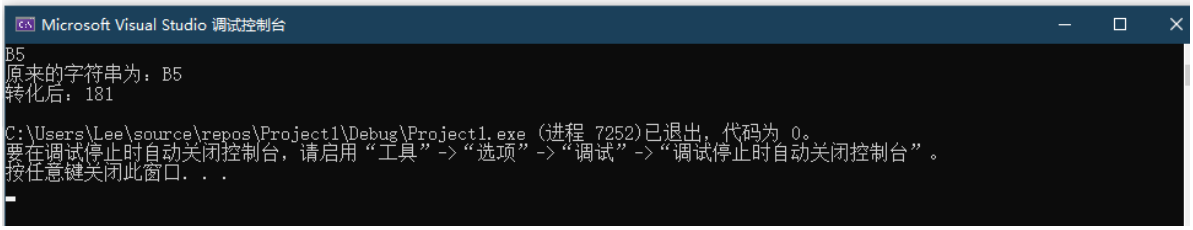
代码如下：

```
#include<iostream>
#include<cstring>
#include<cmath>
using namespace std;
int Transform(string str) {
    int sum = 0;
    int temp = 0;
    bool flag = true;
    int num = 0;

    for (int i = str.length() - 1; i >= 0; i--) {
        switch (str[i])
        {
        case'A':
            temp = 10;
            flag = 0;
            break;
        case'B':
            temp = 11;
            flag = 0;
            break;
        case 'C':
            temp = 12;
            flag = 0;
            break;
        case 'D':
            temp = 13;
            flag = 0;
            break;
        case 'E':
            temp = 14;
            flag = 0;
            break;
        case 'F':
            temp = 15;
            flag = 0;
            break;
```

```
        default:
            break;
        }
        if (flag) {
            temp = (int)(str[i] - '0');
        }
        sum += temp * pow(16, num);
        num++;
    }
    return sum;
}

using namespace std;

int main() {
    string str;
    cin >> str;
    int result = Transform(str);
    cout << "原来的字符串为：" << str<<endl;
    cout << "转化后：" << result << endl;


    return 0;
}
```

输出结果如下：





<3> 编写出判断一个整数是否为素数的**函数**，并求出在2000以内的有十个以上的所有连续的非素数组。

```
#include<iostream>
using namespace std;
#include<cmath>

bool IsPrimeNum(int num)
{
    int m = sqrt(num) + 1;
    for (int i = 2; i < m; i++)
    {
        if (num % i == 0)
        {
            return false;
```

```
        }
    }
    return true;
}

int main()
{
    int a[10];
    int temp = 0;
    for (int i = 3; i < 2000; i++)
    {
        if (!IsPrimeNum(i))
        {
            a[temp] = i;
            temp++;
            if (temp == 10)  //满10个连续非素数输出
            {
                for (int k = 0; k < 10; k++)
                {
                    cout << a[k] << " ";
                }
                cout << endl;
                temp = 0;
                if (!IsPrimeNum(i + 1))
                {
                    i = i - 9;
                    a[temp] = i;
                }
            }
            if (IsPrimeNum(i + 1))
            {
                temp = 0;
            }
        }

    }

    return 0;
}
```

输出结果如下：

```
114 115 116 117 118 119 120 121 122 123
115 116 117 118 119 120 121 122 123 124
116 117 118 119 120 121 122 123 124 125
117 118 119 120 121 122 123 124 125 126
200 201 202 203 204 205 206 207 208 209
201 202 203 204 205 206 207 208 209 210
212 213 214 215 216 217 218 219 220 221
213 214 215 216 217 218 219 220 221 222
294 295 296 297 298 299 300 301 302 303
295 296 297 298 299 300 301 302 303 304
296 297 298 299 300 301 302 303 304 305
297 298 299 300 301 302 303 304 305 306
318 319 320 321 322 323 324 325 326 327
319 320 321 322 323 324 325 326 327 328
320 321 322 323 324 325 326 327 328 329
```

```
321 322 323 324 325 326 327 328 329 330
468 469 470 471 472 473 474 475 476 477
469 470 471 472 473 474 475 476 477 478
510 511 512 513 514 515 516 517 518 519
511 512 513 514 515 516 517 518 519 520
524 525 526 527 528 529 530 531 532 533
525 526 527 528 529 530 531 532 533 534
526 527 528 529 530 531 532 533 534 535
527 528 529 530 531 532 533 534 535 536
528 529 530 531 532 533 534 535 536 537
529 530 531 532 533 534 535 536 537 538
530 531 532 533 534 535 536 537 538 539
531 532 533 534 535 536 537 538 539 540
620 621 622 623 624 625 626 627 628 629
621 622 623 624 625 626 627 628 629 630
662 663 664 665 666 667 668 669 670 671
663 664 665 666 667 668 669 670 671 672
774 775 776 777 778 779 780 781 782 783
775 776 777 778 779 780 781 782 783 784
776 777 778 779 780 781 782 783 784 785
777 778 779 780 781 782 783 784 785 786
798 799 800 801 802 803 804 805 806 807
799 800 801 802 803 804 805 806 807 808
840 841 842 843 844 845 846 847 848 849
841 842 843 844 845 846 847 848 849 850
842 843 844 845 846 847 848 849 850 851
843 844 845 846 847 848 849 850 851 852
864 865 866 867 868 869 870 871 872 873
865 866 867 868 869 870 871 872 873 874
866 867 868 869 870 871 872 873 874 875
867 868 869 870 871 872 873 874 875 876
888 889 890 891 892 893 894 895 896 897
889 890 891 892 893 894 895 896 897 898
890 891 892 893 894 895 896 897 898 899
891 892 893 894 895 896 897 898 899 900
892 893 894 895 896 897 898 899 900 901
893 894 895 896 897 898 899 900 901 902
894 895 896 897 898 899 900 901 902 903
895 896 897 898 899 900 901 902 903 904
896 897 898 899 900 901 902 903 904 905
897 898 899 900 901 902 903 904 905 906
954 955 956 957 958 959 960 961 962 963
955 956 957 958 959 960 961 962 963 964
956 957 958 959 960 961 962 963 964 965
```

注：由于原输出数量过多，所以次处截取一部分

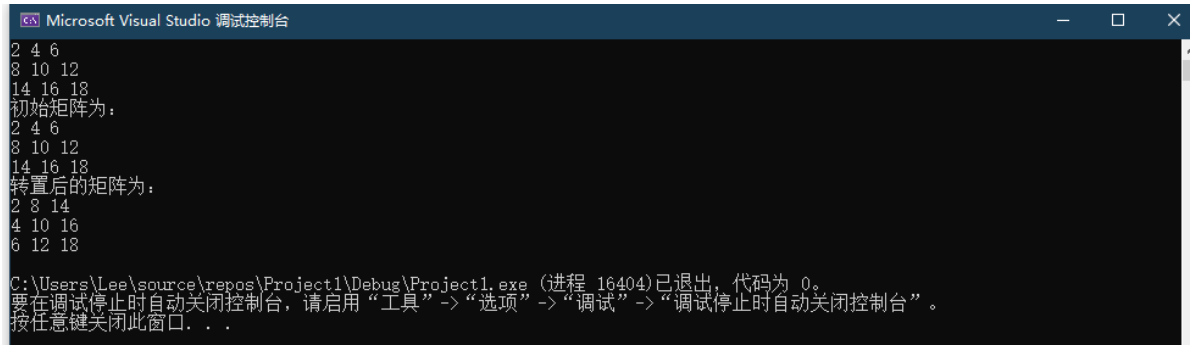
<4>将一个3*3的矩阵转置，用一个**函数**实现。在主函数中输入以下矩阵元素：{2，4，6，8，10，12，14，16，18}。将数组名作为函数参数。函数调用后在主函数中输出已转置的矩阵。

代码如下：

```cpp
#include<iostream>
using namespace std;
```

```cpp
#include<cmath>

void ChangeMatrix(int a[][3])
{
    int temp = 0;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < i; j++)
        {
            temp = a[i][j];
            a[i][j] = a[j][i];
            a[j][i] = temp;
        }
    }
}

int main()
{
    int a[3][3];
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cin >> a[i][j];
        }
    }
    cout << "初始矩阵为：" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    ChangeMatrix(a);
    cout << "转置后的矩阵为：" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

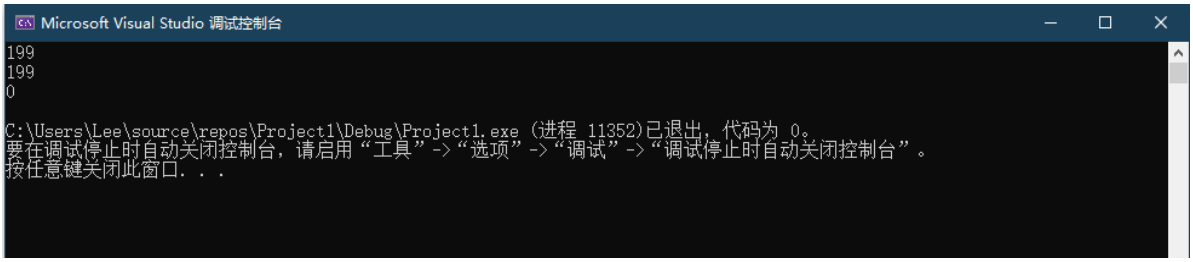
输出如下：

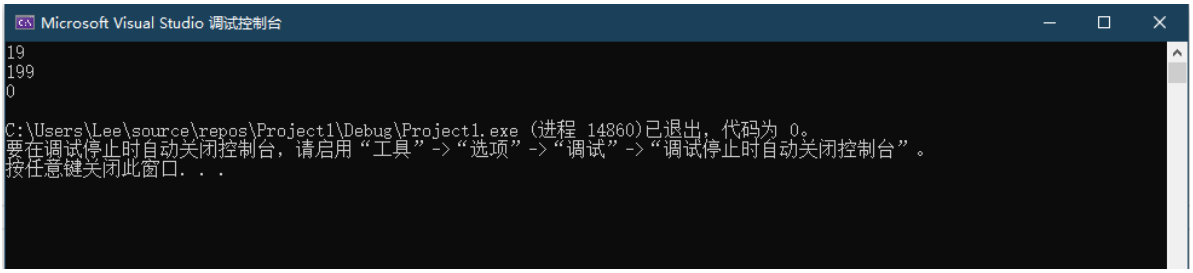


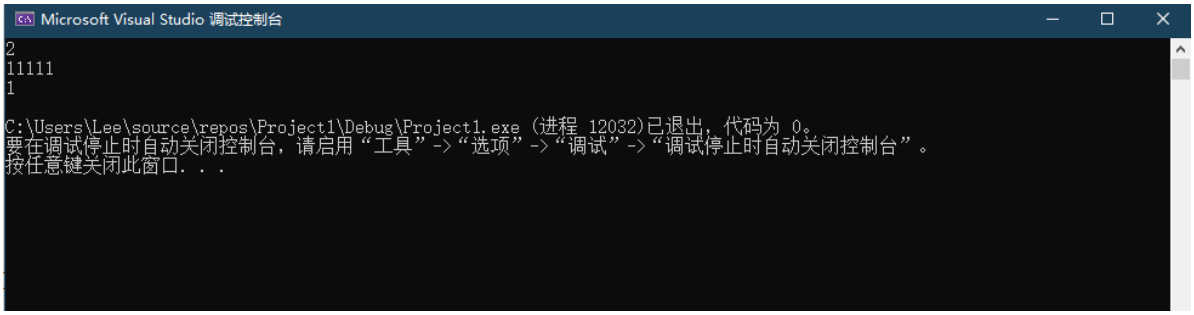
<5>自己写一个strcmp函数，实现两个字符串的比较。当s1<s2时，返回-1；当s1=s2时，返回值= 0；当s1>s2时，返回正数。两个字符串s1,s2由main函数输入，strcmp**函数**的返回值在main函数中输出。

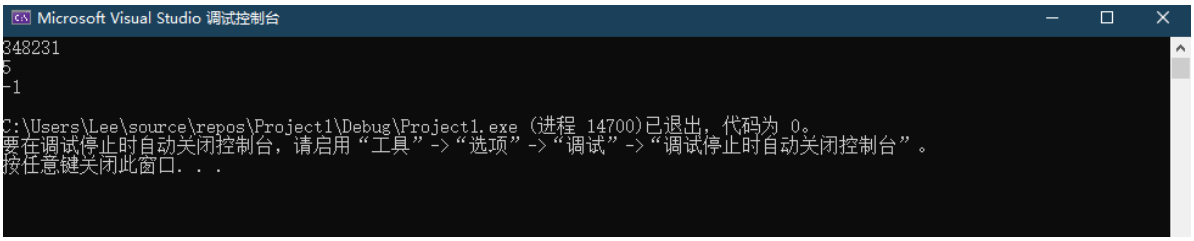
```cpp
#include<iostream>
using namespace std;
#include<cmath>
#include<cstring>
int Strcmp(string s1,string s2)
{
    int minlength = s1.length() > s2.length() ? s2.length() : s1.length();
    for (int i = 0; i < minlength; i++)
    {
        if (s1[i] > s2[i])
        {
            return 1;
        }
        else if (s1[i] < s2[i])
        {
            return -1;
        }
        return 0;
    }
}

int main()
{
    string s1, s2;
    cin >> s1 >> s2;
    cout << Strcmp(s1,s2) << endl;
    return 0;
}
```

输出如下：







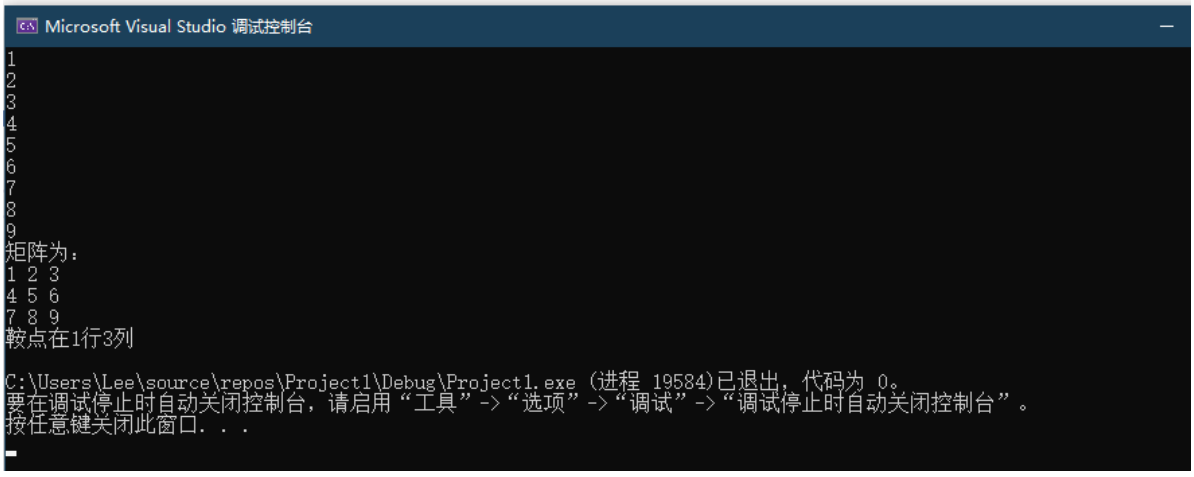


<6> 判断一个二维数组是否有“鞍点”，即某位置上的元素在该行上最大，在该列上最小。如有，输出其行列号和值；若无，给出提示。

```
#include<iostream>
using namespace std;
#include<cmath>

void IsSaddle(int a[3][3])
{
    int k = 0;
    bool flag = 0;
    for (int i = 0; i < 3; i++)
    {
        int max = a[i][0];
        flag = 0;
        for (int j = 0; j < 3; j++)
        {
            if (a[i][j] > max)
            {
                max = a[i][j];
                k = j;
            }
        }
        for (int m = 0; m< 3; m++)
        {
            int min = a[i][k];
            if (a[m][k] <min)
            {
                flag = 1;
            }

```

```
        }
        if (flag == 0)
        {
            cout << "鞍点在" << i+1 << "行" << k+1 << "列" << endl;
            return;
        }
    }
    cout << "无鞍点" << endl;
}
int main()
{
    int a[3][3];
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cin >> a[i][j];
        }
    }
    cout << "矩阵为：" << endl;
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            cout<< a[i][j]<<" ";
        }
        cout << endl;
    }
    IsSaddle(a);
    return 0;
}
```

输出为：



<10> 在某系的成绩登记册中，每个班最多有40个学生，每份成绩表中的成绩信息包括：学号（9位字符），姓名（8位字符），成绩（百分制），备注（20位字符）。设计程序以处理一个班级的成绩信息，包括输入、输出、查询（给定分数以上或以下的学生信息）、按分数排序等。

代码如下：

```cpp
#include<iostream>
using namespace std;
#include<cstring>

void Sort(Student s[], int num)//成绩升序排序
{
    Student temp;
    for (int i = 0; i < num; i++)
    {
        for (int j = i; j < num-1; j++)
        {
            if (s[j].Getscore() > s[j + 1].Getscore())
            {
                temp = s[j];
                s[j] = s[j+1];
                s[i + 1] = temp;
            }
        }
    }
}

void PrintSort(Student s[], int num)//输出按成绩升序排序后的学生信息
{
    for (int i = 0; i < num; i++)
    {
        s[i].PrintInfor();
    }
}

void PrintLower(Student s[], int num, int N)//输出比num成绩低的学生信息
{
    for (int i = 0; i < N; i++)
    {
        if (s[i].Getscore() < num)
        {
            s[i].PrintInfor();
        }
    }
}

void PrintUpper(Student s[], int num, int N)//输出比num成绩高的学生信息
{
    for (int i = 0; i < N; i++)
    {
        if (s[i].Getscore() > num)
        {
            s[i].PrintInfor();
        }
    }
}

class Student
{
public:
    Student() {};//无参构造
    Student(char id[9], char name[8], int score, char remark[20] = { 0 })//
有参构造
    {
```

```
        for (int i = 0; i < 9; i++)
        {
            ID[i] = id[i];
        }
        for (int i = 0; i < 8; i++)
        {
            Name[i] = name[i];
        }
        Score = score;
        for (int i = 0; i < 20; i++)
        {
            Remark[i] = remark[i];
        }
    }
    void PrintInfor()//输出信息
    {
        cout << "学号为: " << endl;
        for (int i = 0; i < 9; i++)
        {
            cout << ID[i];
        }
        cout << endl;
        cout << "姓名为: " << endl;
        for (int i = 0; i < 8; i++)
        {
            cout << Name[i];
        }
        cout << endl;
        cout << "分数为: " << endl;
        cout << Score << endl;
        if (Remark[0])
        {
            cout << "备注: " << endl;
            for (int i = 0; i < 20; i++)
            {
                if (!Remark[i])
                {
                    break;
                }
                cout << Remark[i];
            }
        }
        else
        {
            cout << "无备注" << endl;
        }
    }

    int Getscore()//获取分数
    {
        return Score;
    }

private:
    char ID[9];
    char Name[8];
    int Score;
    char Remark[20];
```

```
};

int main()
{

    return 0;
}
```

4.分析与思考

<1> 在本实验内容2- <10>习题中，若要求成绩部分不仅可以是百分数，而且也可能是五分制（优、良、中、及格和不及格），应如何存储数据以及处理?

解答:

若要求百分制成绩和五分制成绩只能存取一种，可使用结构体

```
Student类中：
private:
    char ID[9];
    char Name[8];
    union Score
    {
        char grade;
        int percent;
    };
    char Remark[20];
```

若百分制成绩和五分制成绩可以同时存取，则可以用结构体或类定义成绩类，再进行类的嵌套

```
class Score
{
public:
    Score() {};
    Score(int percent)
    {
        m_percent = percent;
    }
    Score(char grade, int percent = 0)
    {
        m_grade = grade;
        m_percent = percent;
    }
private:
    char m_grade;
    int m_percent;
};
Student类中：
private:
    char ID[9];
    char Name[8];
    Score score;
    char Remark[20];
```

5.总结

通过这次实验我更加熟悉地掌握了数组的使用，对指针的理解，同时也学到了许多字符相关的操作，可以更好地进行筛选，增删改查等操作，其中还学会了一个将字符转换成整数的小技巧，学会了fgets、strcat等c语言中一些比较实用比较经典的函数。同时我也巩固了之前实验和作业中学习到的各种排序算法，还有求最大公约数的辗转相除法等。同时我在程序设计方法和熟练度上有了更多的提升，也不再是那个拿到程序就不知道如何入手的代码愣头青了。

6.参考资料

(54条消息) strcmp( )函数*henuhhd*的博客-*CSDN*博客strcmp

(54条消息) strcat函数用法*Root_5476*-*CSDN*博客strcat函数用法

(54条消息) C++中 gets（）函数坚持-*CSDN*博客c++ gets函数