算法设计与分析

实

验

报

告

2016年11月

# 目录：

1. 实验一
   1. Problem A: 萌新教程1-1：A+B(1)
   2. Problem B: 萌新教程1-2：A+B(2)
   3. Problem C: 萌新教程1-3：A+B(3)
   4. Problem D: 萌新教程1-4：A+B(4)
   5. Problem E: 萌新教程1-5：A+B(5)
   6. Problem F: 萌新教程1-6：A+B(6)
   7. Problem G: 【枚举】素数判断
   8. Problem H: 填充游戏
   9. Problem I: 字符串比较
   10. Problem J: 男女搭配学习0 0.
2. 实验二
   1. Problem A: 输入你的学号并输出
   2. Problem B: 天下编程 唯快不破
   3. Problem C: 找水果
   4. Problem D: Life Winner TT 的刁难（1）
   5. Problem E: Life Winner TT 的刁难（2）
   6. Problem F: 巧妙排列矩阵
   7. Problem G: 判断回文串
3. 实验三
   1. Problem A: 求十位上的数
   2. Problem B: YW的四边形
   3. Problem C: YW的三角形
   4. Problem D: 【动态规划】TT的药
4. 测试
   1. Problem A: 最近的两个素数
   2. Problem B: 倒数相加
   3. Problem C: 求两个有序数组的中位数
5. 实验总结

# 实验一：

## Problem A: 萌新教程1-1：A+B(1)

#include <iostream>

using namespace std;

int main(){

    int a,b;

    while(cin>>a>>b){

        cout<<a+b<<endl;

    }

    return 0;

}

感想：一个基本的循环输入

---------------------------------------------------------------------

## Problem B: 萌新教程1-2：A+B(2)

#include <iostream>

using namespace std;

int main(){

    int a,b;

    int t;

    cin>>t;

    while(t--){

        cin>>a>>b;

        cout<<a+b<<endl;

    }

    return 0;

}

感想：设置一个变量，存储测试用例数量

---------------------------------------------------------------------

## Problem C: 萌新教程1-3：A+B(3)

#include <iostream>

using namespace std;

int main(){

    int a,b;

    while(cin>>a>>b && a+b){

        cout<<a+b<<endl;

    }

    return 0;

}

感想：循环输入，遇到0退出

---------------------------------------------------------------------

## Problem D: 萌新教程1-4：A+B(4)

#include <iostream>

using namespace std;

int main(){

    int t,sum;

    int n[99] = {0};

    while(cin>>t){

        sum=0;

        for(int i=0;i<t;i++){

            cin>>n[i];

            sum+=n[i];

        }

        cout<<sum<<endl;

    }

    return 0;

}

感想：还是输入

---------------------------------------------------------------------

## Problem E: 萌新教程1-5：A+B(5)

#include <iostream>

using namespace std;

int main(){

    int m;

    int t,sum;

    int n[99] = {0};

    cin>>m;

    while(m--){

        cin>>t;

        sum=0;

        for(int i=0;i<t;i++){

            cin>>n[i];

            sum+=n[i];

        }

        cout<<sum<<endl;

    }

    return 0;

}

感想：没啥大区别

---------------------------------------------------------------------

## Problem F: 萌新教程1-6：A+B(6)

#include <iostream>

using namespace std;

int main(){

    int t,sum;

    int n[99] = {0};

    while(cin>>t && t){

        sum=0;

        for(int i=0;i<t;i++){

            cin>>n[i];

            sum+=n[i];

        }

        cout<<sum<<endl;

    }

    return 0;

}

感想：还行吧

---------------------------------------------------------------------

## Problem G: 【枚举】素数判断

#include <iostream>

#include <cmath>

using namespace std;

int main(){

    int t;

    while(cin>>t){

        int n;

        while(t--){

            bool f=true;

            cin>>n;

            if(n<=1){

                cout<<"No"<<endl;

                f=false;

            }else if(n==2){

                cout<<"Yes"<<endl;

                f=false;

            }else{

                for(int i=2;i<n;i++){

                    if(n%i == 0){

                        cout<<"No"<<endl;

                        f=false;

                        break;

                    }

                }

            }

            if(f!=false){

                cout<<"Yes"<<endl;

            }

        }

    }

    return 0;

}

感想：判断是否是素数，将该数依次除以大于1的数，如果都不能整除，则该数为素数，否则不是。

---------------------------------------------------------------------

## Problem H: 填充游戏

#include <iostream>

#include <string>

using namespace std;

int main(){

    int n;

    string t;

    string str;

    int sz;

    while(cin>>n>>t && n){

        cin.ignore();//为了忽略掉前面的回车或空格，方便后面读取整行字符串

        getline(cin,str);//读取带空格的字符串，以回车结束输入

        sz = str.size();

        if(sz==n){

            cout<<str<<endl;

        }else if(sz<n){

            for(int i=0;i<n-sz;i++){

                cout<<t;

            }

            cout<<str<<endl;

        }else{

            for(int i=sz-n;i<sz;i++){

                cout<<str[i];

            }

            cout<<endl;

        }

    }

    return 0;

}

感想：判断字符串长度，如果大于需要填充的长度，则截取后几位，否则从前面插入相应数量的字符。主要问题在于需要能够输入一串包含空格的字符串，需要用到getline（）,且前面需要调用cin.ignore()忽略掉换行才能正确输入。

---------------------------------------------------------------------

## Problem I: 字符串比较

#include <iostream>

#include <cmath>

#include <string>

using namespace std;

int main(){

    string a,b,c;

    cin>>a>>b>>c;

    string temp;

    if(a>b){

        temp = a;

        a = b;

        b = temp;

        if(b>c){

            temp = c;

            c = b;

            b = temp;

            if(a>b){

                temp = b;

                b = a;

                a = temp;

            }

        }

    }else{

        if(b>c){

            temp = b;

            b = c;

            c = temp;

            if(a>b){

                temp = a;

                a = b;

                b = temp;

            }

        }

    }

    cout<<a<<endl;

    cout<<b<<endl;

    cout<<c<<endl;

    return 0;

}

感想：字符串两两比较即可，最后依次输出结果

---------------------------------------------------------------------

## Problem J: 男女搭配学习0 0.

#include <iostream>

#include <cmath>

#include <string>

using namespace std;

int main(){

    int n;

    string stu[99][3];

    while(cin>>n){

        for(int i=0;i<n;i++){

            cin>>stu[i][0]>>stu[i][1];

            stu[i][2] = "1";

        }

        for(int j=0;j<n/2;j++){

            if(stu[j][0] == "0"){

                for(int k=n-1;k>0;k--){

                    if(stu[k][0] == "1" && stu[k][2]=="1"){

                        stu[k][2] = "0";

                        cout<<stu[j][1]<<" "<<stu[k][1]<<endl;

                        break;

                    }

                }

            }else{

                for(int m=n-1;m>0;m--){

                    if(stu[m][0] == "0" && stu[m][2]=="1"){

                        stu[m][2] = "0";

                        cout<<stu[j][1]<<" "<<stu[m][1]<<endl;

                        break;

                    }

                }

            }

        }

    }

    return 0;

}

感想：输入数据后，将调用双重for循环，依题意，将前面的和后面的依次进行匹配即可。

---------------------------------------------------------------------

# 实验二

## Problem A: 输入你的学号并输出

#include<iostream>

using namespace std;

int main(){

int a;

cin>>a;

if(a<10000000){

cout<<"0";

}

cout<<a;

return 0;

}

感想：学号前面的零需要特别处理

---------------------------------------------------------------------

## Problem B: 天下编程 唯快不破

#include<iostream>

using namespace std;

int main(){

int a;

cin>>a;

for(int i=0;i<a;i++){\

cout<<"Hello WoRld!"<<endl<<endl;

}

return 0;

}

感想：简单for循环即可

---------------------------------------------------------------------

## Problem C: 找水果

#include<iostream>

#include<string>

using namespace std;

int main(){

string fr;

string str;

for(int i=0;i<3;i++){

cin>>fr>>str;

int len = str.size();

int lenf = fr.size();

bool flag = false;

int f=0;

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

if(str[j] == fr[f]){

f++;

if(f==lenf){

cout<<"OK"<<endl;

break;

}

}

}

if(f<lenf){

cout<<"NO"<<endl;

}

}

return 0;

}

感想：双重for循环，依次进行字符匹配即可

---------------------------------------------------------------------

## Problem D: Life Winner TT 的刁难（1）

#include<iostream>

using namespace std;

int main(){

int a[3];

int sum;

int ave;

int temp;

while(cin>>a[0]>>a[1]>>a[2] && a[0]+a[1]+a[2]){

for(int i=2;i>0;i--){

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

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

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

cout<<a[2]-a[0]<<endl;

}

return 0;

}

感想：三个点找距离三点距离最近，只要找到中间那个点即可，简单冒泡排序解得

---------------------------------------------------------------------

## Problem E: Life Winner TT 的刁难（2）

#include<iostream>

using namespace std;

int main(){

int n;

int a[100];

int sum;

int ave;

int temp;

while(cin>>n && n){

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

cin>>a[i];

}

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

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

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

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

if(n%2 != 0){

ave = a[(n-1)/2];

}else{

ave = a[n/2-1];

}

cout<<ave<<endl;

}

return 0;

}

感想：找中位数，如果是奇数，就是中间那个，如果是偶数，求两者的平均数即可，得先排序

---------------------------------------------------------------------

## Problem F: 巧妙排列矩阵

#include<iostream>

using namespace std;

int main(){

int n;

int a[200][200];

int m;

while(cin>>n && n){

a[199][199] = {0};

m = 1;

int down = 0;//列

int left = n-1;//行

int up = n-1;//列

int right = 0;//行

int start = 0;

int endof = n-1;

while(m <= n\*n){

if(m==n\*n){

a[start][start] = m;

break;

}

for(int i=start;i<endof;i++){

a[i][down] = m++;

}

if((m-1)==n\*n){break;}

for(int i=start;i<endof;i++){

a[left][i] = m++;

}

if((m-1)==n\*n){break;}

for(int i=endof;i>start;i--){

a[i][up] = m++;

}

if((m-1)==n\*n){break;}

for(int i=endof;i>start;i--){

a[right][i] = m++;

}

if((m-1)==n\*n){break;}

start++;

endof--;

down++;

left--;

up--;

right++;

}

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

cout<<a[i][0];

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

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

}

cout<<endl;

}

}

return 0;

}

感想：思路大概是，先向下排，再向右，向上，向左，每次排列最后一格留空，交给下个排列，一个回路循环完后，将相应的参数减一，继续下一个循环。到最后，如果是奇数，单独做判断，只需要 填写一个。

---------------------------------------------------------------------

## Problem G: 判断回文串

#include<iostream>

#include<string>

using namespace std;

int main(){

int n;

cin>>n;

string str;

int len;

bool flag;

while(n--){

cin>>str;

len = str.size();

flag = true;

for(int i=0;i<len/2;i++){

if(str[i] != str[len-1-i]){

flag = false;

break;

}

}

if(flag){

cout<<"YES"<<endl;

}else{

cout<<"NO"<<endl;

}

}

return 0;

}

感想：将字符串的前后依次进行匹配即可

---------------------------------------------------------------------

# 实验三

## Problem A: 求十位上的数

#include <iostream>

#include <string>

using namespace std;

int main(){

string a;

int len;

while(cin>>a){

len = a.size();

if(len<=1){

break;

}

cout<<a[len-2]<<endl;

}

return 0;

}

感想：十位上的数简单，但是需要考虑到是负数，还有可能很大很大，所以直接用字符串即可

---------------------------------------------------------------------

## Problem B: YW的四边形

#include <iostream>

#include <string>

using namespace std;

int main(){

int t;

int a[4];

cin>>t;

int i,j,k;

int temp;

int sum;

bool flag;

while(t--){

for(i=0;i<4;i++){

cin>>a[i];

}

flag = true;

for(i=0;i<4;i++){

if(a[i]){

continue;

}else{

flag = false;

}

}

if(flag == false){

cout<<"No"<<endl;

continue;

}

flag = true;

for(i=3;i>0;i--){

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

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

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

if((a[0]+a[1]+a[2])>a[3]){

cout<<"Yes"<<endl;

}else{

cout<<"No"<<endl;

}

}

return 0;

}

感想：先给四条边排序在做对比，如果最长边比其他三条边的和小，就可以。问题在于，别忘了输入的有可能是0，那就不是四边形了。

---------------------------------------------------------------------

## Problem C: YW的三角形

#include <iostream>

#include <algorithm>

#include <string.h>

using namespace std;

bool compare(int a,int b){

return a<b;

}

int a[100005];

int main(){

int t;

cin>>t;

int n;

int i,j,k;

int temp;

bool flag;

while(t--){

cin>>n;

memset(a,0,sizeof(a));

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

cin>>a[i];

}

sort(a,a+n,compare);

flag = false;

for(i=n-1;i>1;i--){

if(a[i]<(a[i-1]+a[i-2])){

cout<<a[i]+a[i-1]+a[i-2]<<endl;

flag = true;

break;

}

}

if(flag == false){

cout<<"OrzYW"<<endl;

}

}

return 0;

}

感想：从很多个数中选出三条边，首先从小到大排序，从大往小取值判断能否组成三角形即可。当然，此处的坑在于，简单的冒泡会超时，可以直接用<algorithm>中的sort()进行排序，别忘了写个函数来配合sort，如果返回的是a<b，则为升序。

---------------------------------------------------------------------

## Problem D: 【动态规划】TT的药

#include <iostream>

#include <string.h>

#include <algorithm>

using namespace std;

int dp[105][9999];

int main(){

int t;

cin>>t;

int i,j,k;

int tmx;

int n;

int c[105];

int v[105];

while(t--){

memset(dp,0,sizeof(dp));

cin>>tmx>>n;

c[0] = v[0] = 0;

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

cin>>c[i]>>v[i];

}

for(i=0;i<=tmx;i++){

dp[0][i] = 0;

}

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

dp[i][0] = 0;

for(j=1;j<=tmx;j++){

if(j<c[i]){

dp[i][j] = dp[i-1][j];

}else{

dp[i][j] = max(dp[i-1][j],dp[i-1][j-c[i]]+v[i]);

}

}

}

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

// for(j=0;j<=tmx;j++){

// cout<<dp[i][j]<<" ";

// }

// cout<<endl;

// }

cout<<dp[n][tmx]<<endl;

}

return 0;

}

感想：简单的O1背包问题，依次填充二维表即可

---------------------------------------------------------------------

# 测试

## Problem A: 最近的两个素数

#include <iostream>

using namespace std;

int main(){

int t;

cin>>t;

int a;

bool flag;

bool two;

int res[2];

while(t--){

cin>>a;

two = false;

if(a==2){

cout<<2<<" "<<2<<endl;

continue;

}

for(int i=a;i>1;i--){

flag = true;

for(int j=2;j<i;j++){

if(i%j == 0){

flag = false;

break;

}

}

if(flag&&!two){

res[0] = i;

//cout<<i;

two = true;

}else if(flag && two){

res[1] = i;

cout<<res[1]<<" "<<res[0]<<endl;

//cout<<" "<<i<<endl;

break;

}

}

}

return 0;

}

感想：从输入的数开始向下查找，依次对每个数进行判断是否为素数，如果是，先保存下来，等找到两个了再从小的开始输出。

---------------------------------------------------------------------

## Problem B: 倒数相加

#include <iostream>

using namespace std;

int main(){

int t;

cin>>t;

int a;

int w[6];

int jin;

while(t--){

w[6] = {0};

int k=0;

cin>>a;

while(a>0){

//cout<<a<<endl;

w[k] = a%10;

a /= 10;

//cout<<w[k]<<"----------"<<a<<"--------"<<k<<endl;

k++;

}

jin = 0;

int sum[7] = {0};

for(int i=k-1,j=0;i>=0;i--,j++){

//cout<<w[i]<<"/"<<w[j]<<"-------"<<endl;

if((w[i]+w[j]+jin)>=10){

sum[i+1] = (w[i]+w[j]+jin)%10;

jin=1;

if(i==0){

sum[0] = 1;

}

}else{

sum[i+1] = (w[i]+w[j])+jin;

jin = 0;

}

}

bool fl = false;

if(sum[0]==1){

cout<<sum[0];

fl = true;

}

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

cout<<sum[i];

}

if(sum[0]==1){

//cout<<endl<<sum[0]<<"/";

k++;

}

cout<<" "<<k<<endl;

}

return 0;

}

感想：考虑到大数，用字符串输入，改写成数组保存，再保存一个倒序的，然后用数组模拟做加法运算即可

---------------------------------------------------------------------

## Problem C: 求两个有序数组的中位数

#include <iostream>

using namespace std;

int main(){

int t;

cin>>t;

int a[20];

int temp;

for(int i=0;i<2\*t;i++){

cin>>a[i];

}

for(int k=2\*t-1;k>0;k--){

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

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

temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

float mid;

mid = ((float)(a[t-1]+a[t]))/2;

cout<<mid<<endl;

return 0;

}

感想：将两个数组合并为一个直接输入，然后排序求中位数即可

---------------------------------------------------------------------

# 实验总结：

算法实验课的题目难度适合刚入门的新手，刚开始都是简单的逻辑运算，后面慢慢加入了贪心和动态规划这类算法，难度有所提升，虽然不能全部解出来，但是整个实验做下来，基础编码能力提升了不少，面对问题时思考也能够更清晰更有信心，理论研究的再多，如果想要有所应用价值，就必须多练习，类似的实验课程的设立还是很有价值很有必要的。