第几天

时间限制: 1000 ms 内存限制: 65536 kb
总通过人数: 1712 总提交人数: 1754

题目描述

给定一个日期，输出这个日期是该年的第几天。

输入

按格式 YYYY/MM/DD 输入日期，保证输入的数据合理。

输出

输出该日期是该年的第几天。

输入样例

```
2018/4/20
```

输出样例

```
110
```

考察知识点

二维数组，闰年判断
难度系数 1

解题思路

这道题即是书上的【例 6-21】，加上主函数的一些输入输出即可；使用二维数组分别记录平年和闰年各个月份的天数，并根据 year 是平年还是闰年而分别使用二维数组中的不同记录

参考代码

```
#include <stdio.h>
#include <stdlib.h>

int day_of_year(int year,int month,int day)
{
    int day_tab[][12] = {{31,28,31,30,31,30,31,31,30,31,30,31},
                         {31,29,31,30,31,30,31,31,30,31,30,31}};
    int leap,i;

    leap = year%400 == 0 || (year%100!=0&&year%4==0);
    for(i=1;i<month;i++)
        day += day_tab[leap][i-1];
    return day;
}

int main()
{
   int year,month,day,t_day;

   scanf("%d/%d/%d",&year,&month,&day);
   t_day = day_of_year(year,month,day);
   printf("%d",t_day);
   return 0;
}
```

傻傻 Aqi 的成绩统计函数

时间限制: 1000 ms 内存限制: 65536 kb
总通过人数: 1266 总提交人数: 1391

题目描述

傻傻 Aqi 和 Alice 受命用程序帮班里统计 C 语言课成绩。可是他们却吵架了，因为 Alice 提出指针的思路之后，傻傻 Aqi 不采纳还一意孤行，结果他设计的函数只有一个返回值故不能正常工作。现在傻傻 Aqi 知道错了，他想带着用了指针的函数去向 Alice 负荆请罪，请大家来帮他吧，以帮助他们尽快和好哟~

设计一个函数，从标准输入上读入数量不定的成绩。统计全班成绩的总数量、 90 分及以上成绩的个数、60 分及以上成绩的个数和全班成绩的平均分，并返回这四个数给主程序。编写主程序调用此函数完成统计功能。

输入

单组数据输入， 输入数量不定（至少 1 个）的非负整数，为班里成绩同学的成绩。

输出

对于该组输入，
第一行输出全班成绩的总数量 nn，
第二行输出 90 分及以上成绩的个数 aa，
第三行输出 60 分及以上成绩的个数 bb，
第四行输出全班成绩的平均分 pp（保留 2 位小数）。

输入样例

```
10 20 30 60 90 100
```

输出样例

```
6
2
3
51.67
```

考察知识点

```
数组或指针
难度系数 2
```

解题思路

这道题改编自第 7 章 PPT 的例 7.2。旨在告诉同学们，此前我们了解到函数最多只有一个返回值，传进来的参数也只在函数内部改变，函数结束后，其值不会改变。但是运用指针，可以直接改变传进来的参数，非常方便。而之所以此前学的数组参数在函数中也有相同的性质，也是基于指针的原理，因为数组名就是一个指向数组首地址的指针。

代码

```
#include <stdio.h>

void data_stat(int *,int *,int *,float *);

void data_stat(int *p_num,int *p_90,int *p_60,float *avg){
	int v,sum=0;
	while(scanf("%d",&v)!=EOF){
		(*p_num)++;
		if((v>90)||(v==90)){
			(*p_90)++;
		}
		if((v>60)||(v==60)){
			(*p_60)++;
		}
		sum=sum+v;
	}
	*avg=(float)sum/(*p_num);
}

int main(){
	int n=0,a=0,b=0,*np=&n,*ap=&a,*bp=&b;
	float p=0.0,*pp=&p;

	data_stat(np,ap,bp,pp);
	printf("%d\n%d\n%d\n%.2f",n,a,b,p);
	return 0;
}
```

623 不知道星期几

时间限制：1000ms　内存限制：65536kb

通过率：1503/1559 (96.41%)　正确率：1503/2334 (64.40%)

题目描述

已知本月有 x 天，第 y 天是星期 n，求下月 k 日是星期几。

输入

一行，四个整数，分别是 x，y，n，k，其中 n=0 表示星期日。

输出

一行字符串，表示星期几，要求首字母大写。

输入样例

```
31 1 0 20
```

输出样例

```
Monday
```

考察知识点

二维数组
难度系数 2

解题思路

本题是教材和 ppt 上的例 6-23，思路就是用已知的 4 个值来求指定日期是星期几，然后用字符串数组作为输出，只需要将例题中的代码稍微修改即可通过，较简单。具体操作见代码。

标程

```
#include <stdio.h>
char day_name[][12] =
{
        "Sunday",
        "Monday",
        "Tuesday",
        "Wednesday",
        "Thursday",
        "Friday",
        "Saturday"
};

int weekday(int, int, int, int);
int main(){
        int x, y, n, k, m;
        x = 1; y = 0; n = 31;
        scanf("%d%d%d%d",&n,&x,&y,&k);
        m = weekday(x, y, n, k);
        printf("%s\n", day_name[m]);
        return 0;
}
int weekday(int x, int y, int n, int k){
        return (n - x +k +y)%7;
}
```

阿狄的加密文件

时间限制：1000ms　内存限制：65536kb

通过率：1076/1281 (84.00%)　正确率：1076/3941 (27.30%)

题目描述

阿狄得到了一份加密的文件，为了得到其中的信息他决定从最复杂也就是最长的一行开始入手，忙于破解加密方式的阿狄决定拜托你们，找到最长的一行的长度和内容。

输入

多行字符串输入，每行文字只包含 a-z 小写字母，行数小于 1000，行长小于 200。

输出

一行，包括一个整数（表示这个文件中最长行长度）和最长字符串，格式与样例一致 ，若有多个相同长度的最长行，则输出第一个最长行。

输入样例

```
aaa
bbbb
cc
```

输出样例

```
4:bbbb
```

题解：因为不同换行符的问题导致 gets 不能正确的使用，但全字母的字符串用 scanf 也可以读入。用两个指针指向字符数组，分别表示最长行字符串和当前字符串，当前行的长度更长时交换指针的目的地址。

```
#include<stdio.h>
#include<string.h>
int main()
{
    char buf1[220],buf2[220];
    char *lg = buf2,*now = buf1,*tmp;
    int ml = 0,len;
    while(scanf("%s",now)!=EOF)
    {
        len = strlen(now);
        if(len>ml)
        {
            ml = len;
            tmp = lg;
            lg = now;
            now = tmp;
        }
    }
    printf("%d:%s",ml,lg);
```

}

子串逆置

时间限制: 1000 ms 内存限制: 65536 kb
总通过人数: 640 总提交人数: 1003

题目描述

从标准输入上读入以空格分隔的字符串 s 和 t，将 s 中与 t 匹配的所有子串逆置后再输出 s，当 s 中无与 t 匹配的子串时直接输出 s。已经匹配的字符不会再重复匹配。

输入

以空格分隔的字符串 s 和 t。s，t 长度小于 100。

输出

输出逆置后的 s。

输入样例

```
helloworld wor
```

输出样例

```
hellorowld
```

考察知识点：字符串处理
解题思路： 读入两个字符串 s 和 t，使用标准库的 strstr()判断 s 中是否包含 t。但是要匹配所有的 t，且已经匹配的字符不会再重复匹配，于是可以利用 strstr()返回的指针位置与 t 的长度来移动指向 s 的指针，直至匹配不到 t。

代码：

```
#include <stdio.h>
#include <string.h>

void rev(char* first, char* last) {
        int tmp;
        while (first < last) {
          tmp = *last;
          *last = *first;
          *first = tmp;
          first++, last--;
        }
}
int main() {
        char str[BUFSIZ], substr[BUFSIZ], *p = str;

        scanf("%s%s", str, substr);
        while ((p = strstr(p, substr)) != NULL) {
          rev(p, p + strlen(substr) - 1);
          p = p + strlen(substr);
        }
        puts(str);
    return 0;
}
```

Tarpe 酋长的图像旋转器

时间限制：1000ms　内存限制：65536kb

通过率：1036/1148 (90.24%)　正确率：1036/1827 (56.70%)

题目描述

其实图片就是一个简单的矩阵而已，Tarpe 酋长想让这个矩阵顺时针旋转 90 度，输出旋转后的结果。

输入

第一个数 n，
表示 n*n 的矩阵。
(保证 0<n<100
接下来 n 行，每行 n 个数，为矩阵的元素。

输出

旋转后的矩阵

输入样例

```
3
1 2 3
4 5 6
7 8 9
```

输出样例

```
7 4 1
8 5 2
9 6 3
```

考察知识点

二维数组，难度系数 6

解题思路：

思路 1：

很多同学都是创建一个新的二维数组，将元素坐标作变换，然后复制到新的二维数组，输出新的二维数组，这种方法有两个缺点，一个是需要分类讨论，第二是占用了额外的空间。

思路 2：

原地变换：不开新的二维数组，直接交换数组中对应位置的元素，这个方法比较有技巧性，每次修改了四个位置的元素，这样就可以减少遍历的时间，也节约了空间，具体做法见代码。

下面展示的是思路 2 的代码：

```
#include<stdio.h>
#include<stdlib.h>
int matrix[105][105];
int n = 0;
void rotate(int n) {
    for (int i=0; i<n/2; ++i)
    {
```

```
            for (int j=i; j<n-1-i; ++j)
            {
                int z = matrix[i][j];
                matrix[i][j] = matrix[n-j-1][i];
                matrix[n-j-1][i] = matrix[n-i-1][n-j-1];
                matrix[n-i-1][n-j-1] = matrix[j][n-i-1];
                matrix[j][n-i-1] = z;
            }
        }
}
int main()
{
        //freopen("in3.txt","r",stdin);
        //freopen("out3.txt", "w", stdout);
        scanf("%d", &n);
        for(int i=0; i<n; i++)
        {
            for(int j=0; j<n; j++)
            {
                scanf("%d",&matrix[i][j]);
            }
        }
```

```
        rotate(n);
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                printf("%d ",matrix[i][j]);
            }
            printf("\n");
        }
        //fclose(stdin);
        //fclose(stdout);
}
```

Tarpe 酋长的超级计算器

时间限制：1000ms　内存限制：65536kb

通过率：413/718 (57.52%)　正确率：413/2001 (20.64%)

题目描述

酋长打算出一道水题来改善一下气氛。。。
也许 1∗1=11∗1=1 是个不错的选择？

输入

两行，每行一个数字，分别是乘数 a 和 b。
其中 a，b 为非负整数，a<=10200a<=10200 且 b<=10200b<=10200。

输出

输出这两个数字的乘积

输入样例

```
2
3
```

输出样例

```
6
```

HINT

```
long long is never enough
试试看用字符串保存大数字？不要忘了考虑数字顺序问题和进位问题哦。
```

考察知识点

```
字符串模拟计算
难度系数 6
```

坑点：

当乘数为 0 时需要特判。

解题思路：

经典问题，大数乘法或者说是高精度乘法。具体来说就是用字符串来模拟数字的计算，用来弥补 longlong 精度不够的问题。

1. 我们用字符串来存储数字的输入。
2. 将字符串中的字符转换为数字，并且倒序保存在数组中（想一想乘法的竖式原理？）
3. 将乘数各个位上的数字与另一个各个位上的数字相乘，先乘起来，后面统一进行进位
4. 进位
5. 删除 0 的前缀
6. 倒序输出

AC 代码：

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

char s1[205],s2[205];
int ans[410];
char s[410];
int arr[410];

int main()
{
    //freopen("in3.txt","r",stdin);
```

```
//freopen("out3.txt","w",stdout);
scanf("%s",&s1);
scanf("%s",&s2);
int l1 = strlen(s1);
int l2 = strlen(s2);



for(int i=0;i<l1+l2;i++)
{
	ans[i] = 0;
}

for(int i=0;i<l1;i++){
	int carry = 0;
	int n1 = (int)(s1[l1-i-1]-'0');
	for(int j=0;j<l2;j++){
		int n2 = (s2[l2-j-1]-'0');
		int sum=n1*n2 + ans[i+j] + carry;
		carry = sum/10;
		ans[i+j] = sum%10;
	}
	if(carry>0)
```

```
                ans[i+l2]+=carry;
        }


        int start = l1+l2-1;
        while(ans[start] == 0)start--;
        if(start <0)printf("0\n");
        else{
                int k =0;
                for(int i=start;i>=0;i--)
                {
                        s[k++] = (char)(ans[i]+'0');
                }
                s[k] = '\0';
                printf("%s",s);
        }
        //fclose(stdin);
        //fclose(stdout);
}
```

Tarpe 酋长的 IP 解析

时间限制：1000ms　内存限制：65536kb

通过率：206/432 `(47.69%)`　正确率：206/1316 `(15.65%)`

题目描述

现在的 IP 地址一般有两种格式，IPv4 和 IPv6。

IPv4 地址是以十进制表示的标准格式，由四个十进制数字组成，每个十进制数字范围从 0 到 255，用点（“.”）分隔，例如 `172.16.254.1`;
另外，IPv4 中的前导零是无效的。例如，地址 `172.16.254.01` 无效。
IPv6 地址被表示为八组四位十六进制数字。这些组由冒号分隔（“:”）。例如，地址 `2001:0db8:85a3:0000:0000:8a2e:0370:7334` 是有效的。此外，我们可以省略四位十六进制数字中的一些前导零，而且地址不区分大小写字母，因此 `2001:db8:85a3:0:0:8A2E:0370:7334` 也是有效的 IPv6 地址（省略前导零和使用大写）。但是，我们不会使用两个连续的冒号（::)来替换单个空组的零值的连续组。例如，`2001:0db8:85a3 :: 8A2E:0370:7334` 是无效的 IPv6 地址。
另外，IPv6 中额外的前导零（超出 4 位）也是无效的。例如，地址 `02001:0db8:85a3:0000:0000:8a2e:0370:7334` 无效。
Tarpe 酋长希望你能判断该地址是否有效，如果有效，还希望判断一下是 IPv4 还是 IPv6 地址。

输入

一行字符串表示 IP 地址（长度小于 50）。
只有一组输入。
输入的字符串只包含数字，大小写字母，‘：’，‘.’

输出

输出一行判断结果，如果无效，输出 `Invalid`，如果有效，输出 `IPv4` 或者 `IPv6`

输入样例

```
172.16.254.1
2001:0db8:85a3:0:0:8A2E:0370:7334
256.256.256.256
```

输出样例

```
IPv4
IPv6
Invalid
```

考察知识点

```
字符串处理，难度系数 8
```

坑点：

1. IPv4 的数字是有范围的，0~255
2. IPv4 的前导零无效
3. IPv6 的地址是不区分大小写的
4. IPv6 的地址中的前导零无效
5. IPv6 中不应存在两个连续的冒号

解题思路：

这道题考察了复杂的字符串处理，首先你需要确定是十六进制还是八进制，然后再对 IPv4 和 IPv6 分别进行处理，助教的代码比较复杂，可以参考李志璞同学的代码（虽然有点面向数据编程的意思）。

助教版代码（C++）：

```
#include<iostream>
#include<string.h>
#include<string>
```

```
#include<cstdio>
#include<strings.h>
#include<sstream>
using namespace std;

const string validIPv6Chars = "0123456789abcdefABCDEF";

bool isValidIPv4Block(string& block) {
    int num = 0;
    if (block.size() > 0 && block.size() <= 3) {
        for (int i = 0; i < block.size(); i++) {
            char c = block[i];
            // special case: if c is a leading zero and there are characters left
            if (!isalnum(c) || (i == 0 && c == '0' && block.size() > 1))
            return false;
            else {
            num *= 10;
            num += c - '0';
            }
        }
        return num <= 255;
    }
```

```
        return false;
    }

    bool isValidIPv6Block(string& block) {
        if (block.size() > 0 && block.size() <= 4) {
            for (int i = 0; i < block.size(); i++) {
                char c = block[i];
                if (validIPv6Chars.find(c) == string::npos)
                    return false;
            }
            return true;
        }
        return false;
    }

    string validIPAddress(string IP) {
        string ans[3] = {"IPv4", "IPv6", "Invalid"};
        stringstream ss(IP);
        string block;
        // ipv4 candidate
        if (IP.substr(0, 4).find('.') != string::npos) {
            for (int i = 0; i < 4; i++) {
```

```
            if (!getline(ss, block, '.') || !isValidIPv4Block(block))
                return ans[2];
            }
            return ss.eof() ? ans[0] : ans[2];
        }
        // ipv6 candidate
        else if (IP.substr(0, 5).find(':') != string::npos) {
            for (int i = 0; i < 8; i++) {
            if (!getline(ss, block, ':') || !isValidIPv6Block(block))
                return ans[2];
            }
            return ss.eof() ? ans[1] : ans[2];
        }

        return ans[2];
}

int main()
{
        //ifstream in("in5.txt");
        //ofstream out("out5.txt");
        string IP;
```

```
    cin>>IP;
    string ans = validIPAddress(IP);
    cout<<ans<<endl;
}
```

同学优秀代码：

```
#include <stdio.h>
#include <string.h>
char a[50];
int main()
{
    int n,j=0,k=0,i,c1=0,key=0;
    gets(a);
    n=strlen(a);
    for(i=0;i<n;i++)
    {
        if(a[i]==':') j++;
        if(a[i]=='.') k++;
    }
    if(j>0&&j<8&&k==0)//十六进制
    {
        for(i=0;i<n;i++)
```

```
        {
            if((a[i]>'F'&&a[i]<='Z')||(a[i]>'f'&&a[i]<='z')){c1=1;break;}
            if(a[i]!=':')
            {
                key++;
                if(key>4){c1=1;break;}
            }
            if(a[i]==':'){key=0;}
              if(a[i]==':'&&a[i-1]==':'){c1=1;break;}
              if((a[i]>'F'&&a[i]<='Z')||(a[i]>'f'&&a[i]<='z')){c1=1;break;}
        }
        if(c1==0) printf("IPv6");
        else printf("Invalid");
}
else if(j==0&&k>0&&k<4)// 十进制
{
        for(i=0;i<n;i++)
        {
            if(a[i]>'9'){c1=2;break;}
            if(a[i]=='0'&&(i==0||a[i-1]=='.')){c1=2;break;}
            if(a[i]=='2'&&a[i+1]=='5'&&a[i+2]>'5'){c1=2;break;}
            if(a[i]=='2'&&a[i+1]>'5'&&a[i+2]>='0'){c1=2;break;}
```

```
                if(a[i]>'2'&&a[i+1]>='0'&&a[i+2]>='0'){c1=2;break;}


if(a[i]>='0'&&a[i+1]>='0'&&a[i+2]>='0'&&a[i+3]>='0'){c1=2;break;}
            }
            if(c1<2) printf("IPv4");
            else printf("Invalid");
        }
        else printf("Invalid");
        return 0;
}
```

水水的字符串匹配

时间限制：1000ms　内存限制：65536kb

通过率：662/810 (81.73%)　正确率：662/1567 (42.25%)

题目描述

给定两个不同长度的字符串，输出第一个字符串中所含第二个字符串的数量（允许重叠计算）。

输入

共两行。每行一个字符串，保证其中第一个字符串长度不小于第二个字符串，且两字符串长度均不大于 100。

输出

一个整数，表示字符串一中含字符串二的数量。

输入样例

```
abbabbcd
abb
```

输出样例

```
2
```

考察知识点

```
字符串处理
难度系数 6
```

解题思路

本题的思路就是读入两个字符串，然后遍历第一个字符串的每一个字符，并判断以当前字符为起点是否可以匹配，最后输出即可，思路非常朴素，只需要注意允许重叠计算即可。具体操作见代码。

标程

```
#include <stdio.h>
#include <string.h>

char A[102],B[102];
int count(int a,int b)//a is the start A , b is B's length
{
	int m;
	for(m=0;m<=b-1;m++)
	{
		if(A[a+m]!=B[m]) return 0;
	}
	return 1;
}
int main()
{
	int lenA,lenB;
```

```
    scanf("%s%s",&A,&B);
    lenA=strlen(A);
    lenB=strlen(B);
    int p,num=0;
    for(p=0;p<=lenA-lenB;p++)
    {
        num+=count(p,lenB);
    }
    printf("%d\n",num);
}
```

橙橙的烦恼

时间限制：1000ms　内存限制：65536kb

通过率：539/697 (77.33%)　正确率：539/1753 (30.75%)

题目描述

橙橙是个小糊涂，眼神不是很好，他把手放在键盘上的时候，稍不注意就会往右错一位。这样，输入 Q 会变成输入 W，输入 J 会变成输入 K 等等,但空格的输入还是正常的。 亲爱的同学们，你能帮橙橙把打错的字符串还原吗？

输入

一个错位后敲出的字符串（有可能出现数字和键盘上的各个符号，所有字母均大写，且保证输入合法，即一定是错位之后的字符串，例如输入中不会出现大写字母 A)。
不包含键盘的功能键

输出

输出橙橙本来想打出的句子。

输入样例

```
O S, GOMR YPFSU/
```

输出样例

```
I AM FINE TODAY.
```

考察知识点

字符串处理
难度系数 6

解题思路

本题单纯用 if 和 switch 语句写也可以，但是太过于麻烦。

那么一个比较好的办法是使用常量数组，构造一个数组将键盘上的每个数字、符号、字母按照键盘排列顺序记录下来，使得对于每一个打错的字符，它前面一个的字符就是正确的字符，如此便于查找和搜索。

参考代码如下：

```
#include<stdio.h>
char s[]="`1234567890-=QWERTYUIOP[]\\ASDFGHJKL;'ZXCVBNM,./";
int main()
{
	int i,c;
	while((c=getchar())!=EOF)
	{
		for(i=1;s[i]&&s[i]!=c;i++);
		if(s[i]) putchar(s[i-1]);
		else putchar(c);
	}
	return 0;
}
```