A 来输出字符串吧!

| 难度 | 考点 |

| ---- | ------- |

| 1 | 循环、转义符 |

题目分析

只需要按照HINT里的提示，将 `\`、`"`、`'`、`%` 用对应的转义符输出即可。

示例代码

```
#include<stdio.h>

int main(){

    printf("\"%%o\'\\n/\'o%%\"");

    return 0;

}
```

B 字符也是数

| 难度 | 考点 |

| ---- | ------- |

| 1 | ascii码 |

题目解析

C语言中，所有可见字符都是以ascii码的形式存储在`char`类型的数据当中的。`char`类型的数据本身的大小为1字节，能表示$[-256, 255]$范围内的整数，足够用来存储所有ascii码。当然也有表示不可见字符的ascii码，比如换行符`\n`是10，回车符`\r`是13。

在打印ascii码的实际值的时候，为了编程规范，先将存储字符的`char`类型变量转换为`int`类型变量，再使用`printf`和格式控制串`%d`进行输出。

示例代码

```
#include <stdio.h>

int main(){

    char c;

    scanf("%c", &c);

    int n = (int) c;

    printf("%d is the ASCII code for %c.", n, c);
```

```
    return 0;
}
```

C 一元二次方程 真 便 宜

题目分析

本题考查判断语句的使用、小数位数的保留、`sqrt()`函数的使用

示例代码

```
#include <math.h>
#include <stdio.h>
int main()
{
    double a, b, c;
    scanf("%lf%lf%lf", &a, &b, &c);
    double d = b * b - 4 * a * c;
    if (d < 0)
        printf("no real roots\n");
    else if (d == 0)
        printf("%.6f %.6f\n", -b / 2 / a, -b / 2 / a);
    else
    {
        double r1 = (-b + sqrt(d)) / 2 / a;
        double r2 = (-b - sqrt(d)) / 2 / a;
        if (r1 < r2)
            printf("%.6f %.6f\n", r1, r2);
        else
            printf("%.6f %.6f\n", r2, r1);
    }
}
```

D 火仙草

| 难度 | 考点 |

| :--: | :--------: |

| 2 | 循环、判断 |

解题思路

本题直接使用 `for` 循环遍历范围的每个数字是否是火仙草数即可，需要注意：

- 计数用的变量需要初始化：`count=0`

- 不要重复统计：`if - else if`

样例代码

```
#include<stdio.h>
int main()

{

int i,a,b,c,d,e,f,n,count=0;

scanf("%d",&n);

for(i=1000;i<=n;i++)

{

a=i/1000;//千位

b=(i%1000)/100;//百位

c=(i%100)/10;//十位

d=i%10;//个位

e=i/100;//前两位

f=i%100;//后两位

if(a*a*a*a+b*b*b*b+c*c*c*c+d*d*d*d==i) count++;

else if(e*e+f*f==i) count++;

else if(i%666==0) count++;

else if(i%888==0) count++;

}

printf("%d",count);

return 0;

}
```

E **简单后缀运算**

| 难度 | 考点 |

| ---- | ---------- |

| 2 | 字符、判断 |

题目分析

本题考察EOF结束的多组数据输入以及判断；

```
int a,b;

char ch;

scanf("%d%d%c",&a,&b,&ch);  //语句1

scanf("%d%d %c",&a,&b,&ch); //语句2
```

以上两个语句的区别是，在读入第2个整数赋值给变量b之后，

* 语句1会紧接着把后面一个字符读入给变量ch，即使这个字符是空格等空白符；

* 语句2会跳过空白符直到遇到非空白符才进行读取，然后赋值给变量ch；

也就是说，对于下列输出，两个语句运行后的结果是这样的：

```
// input: 3 4 +

// 语句1:a=3, b=4, ch=' '(空格字符)

// 语句1:a=3, b=4, ch='+'(加号字符)
```

考虑了这个点之后，代码就相对好写了

参考代码

```
// C1-E 简单后缀运算

// author: Song You

// 2022-3-5

\#include <stdio.h>

int main()

{

    int a,b;

    char ch;

    while(scanf("%d%d %c",&a,&b,&ch) != EOF)

    {
```

```
        if(b == 0 && (ch == '/' || ch == '%'))
        {
            printf("Runtime Error (SIGFPE)\n");
        }
        else
        {
            if(ch == '+')
                printf("%d\n", a+b);
            else if(ch == '-')
                printf("%d\n", a-b);
            else if(ch == '*')
                printf("%d\n", a*b);
            else if(ch == '/')
                printf("%d\n", a/b);
            else
                printf("%d\n", a%b);
        }
    }
    return 0;
}
/*
// 如下实现的逻辑性更强，可以画一个决策树来理解
#include <stdio.h>
int main()
{
    int a,b;
    char ch;
    while(scanf("%d%d %c",&a,&b,&ch) != EOF)
    {
```

```
        if(ch == '+')
            printf("%d\n", a+b);
        else if(ch == '-')
            printf("%d\n", a-b);
        else if(ch == '*')
            printf("%d\n", a*b);
        else
        {
            if(b == 0)
                printf("Runtime Error (SIGFPE)\n");
            else
            {
                if(ch == '/')
                    printf("%d\n", a/b);
                else
                    printf("%d\n", a%b);
            }
        }
    }
    return 0;
}
```

F 字符变变变

| 难度 | 考点 |

| ---- | ---------- |

| 2 | 字符，ASCII码 |

题目分析

参考课件小写字母转换成大写字母的写法，大写字母转换成小写字母同理；

这里提供一种不用记ASCII码的写法，由于字符实际上是个小整数，也可以像其他整数一样进行数值运算，所以假如ch='A'，那么ch-'A'+'a'就是'a'，对于一般的大写字母字符该表达式也可以将其转换成相应的小写字母字符，另一半同理

参考代码1

```
#include <stdio.h>
int main()
{
    char ch;
    while ((ch = getchar()) != EOF)
    {
        if (ch >= 'a' && ch <= 'z')
            putchar(ch - 'a' + 'A');
        else if (ch >= 'A' && ch <= 'Z')
            putchar(ch - 'A' + 'a');
        else
            putchar(ch);
    }
    return 0;
}
```

提供一种采用 <ctype.h> 库的写法

参考代码2

```
// Copyright: 张京泽
#include <stdio.h>
#include <ctype.h>
int main()
{
    char c;
    while (scanf("%c", &c) != EOF)
    {
        if (isupper(c))
            printf("%c", tolower(c));
```

```
        else if (islower(c))
            printf("%c", toupper(c));
        else
            printf("%c", c);
    }
    return 0;
}
```

G 统计统计成绩

| 难度 | 考点 |

| ---- | ---- |

| 2 | 数组 |

题目分析

数组的简单应用，注意数组的下标是从0～n-1，另外是**大于等于**，不是大于

参考代码

```
#include <stdio.h>
int n;
int score[1005];
double ave;
int main(void)
{
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &score[i]);
        ave += score[i];
    }
    ave = ave / n;
    double sum = 0;
```

```
    int all = 0;
    for (int i = 0; i < n; i++)
    {
        if (score[i] >= ave)
            sum += score[i], all++;
    }
    printf("%.2f\n", sum / all);

    return 0;
}
```

H **甲贺忍蛙的名次预测**

解法一：课件解法

仿照课件直接写代码即可。

```
#include<stdio.h>

int main()
{
    int a,b,c,d,e,f,g,h;
    while(scanf("%d%d%d%d%d%d%d%d",&a,&b,&c,&d,&e,&f,&g,&h)!=EOF)
    {
        if((a==1)+(b==2)+(c==3)+(d==4)+(e==5)+(f==6)+(g==7)+(h==8)>=5)
        {
            printf("Kouga!\n");
        }
        else
        {
            printf("Gek...\n*
        }
    }
}
```

解法二：计数解法

借助一维数组下标，构造一个计数变量进行计数。

```
#include<stdio.h>

int main()
{
    int ranking[8];
```

```
 while(scanf("%d%d%d%d%d%d%d%d",&ranking[0],&ranking[1],&ranking[2],&ranking[3],
&ranking[4],&ranking[5],&ranking[6],&ranking[7])!=EOF)
    {
        int count=0;
        int i;
        for(i=0;i<8;i++)
        {
            if(i+1==ranking[i])
            {
                count++;
            }
        }
        printf(count>4?"Kouga!\n":"Gek...\n");
    }
}
```

I 时刻2.0

| 难度 | 考点 |

| :--: | :----------: |

| 3 | 多组数据、换行符处理、数据处理 |

解题思路

本题考查的重点是一行需要读入的首个内容是字符类型时，如何处理上一行末的换行符。

`scanf` 在对输入进行匹配时，除了按照类型匹配外也可以处理固定字符，比如本题示例代码用 `%d:%d` 来跳过时刻中间的冒号。如果需要跳过的字符是空白符，例如 ` `（空格）、`\n`（换行符 Line Feed, LF）、`\r`（回车符 Carriage Return, CR）、`\t`（制表符）等等，可以使用任一空白符表示跳过连续的若干空白符，本题示例代码用 `"%d "` 来处理行末的换行符。

部分同学在本地测试可能发现无法结束输入的问题，这是因为 `scanf` 在跳过空白符时会跳过连续的所有空白符只到下一个非空白符，系统无法判断空白符是否结束，也就不能结束输入，这时需要手动结束输入流，Windows系统的文件结束符（End of File, EOF）是 `Ctrl`+`Z`，Linux系统使用`Ctrl`+`D`，MacOS系统使用`Control`+`D`，XCode环境先按`Control`+`Q`再按`Control`+`D`，在新的一行使用即可结束输入流。

当然，`scanf` 除了上述匹配规则外，还有指定输入范围等很多匹配方式，不同的匹配类型也有不同的辅助功能，例如 `%d` 可以跳过数字前的空白符，感兴趣的同学可以自行了解。

示例代码

```
#include<stdio.h>

int main()

{

int h1,m1,h2,m2,n;

char a1,a2,k;

scanf("%d ",&n);
```

```
while(n--)//多组数据读入

{

scanf("%c%c %d:%d ",&a1,&k,&h1,&m1);

scanf("%c%c %d:%d ",&a2,&k,&h2,&m2);

if(h1==12) h1-=12;//12小时制转24小时制

if(h2==12) h2-=12;

if(a1=='P') h1+=12;

if(a2=='P') h2+=12;

if(h1<h2) printf("2\n");//判断早晚关系

else if(h1>h2) printf("1\n");

else

{

if(m1<m2) printf("2\n");

else if(m1>m2) printf("1\n");

else printf("Same Time\n");

}

}

return 0;

}
```

当你按下回车之后——关于\r\n那些事儿

当你按下回车键之后，光标移动到了下一行，但是文件中发生了什么呢?

不同的操作系统对此的反应也不同:

| 操作系统 | 换行内容 | 控制代码 |

| :------: | :------: | :------: |

| Windows | \r\n | CRLF |

| Linux | \n | LF |

| MacOS | \r | CR |

在计算机还没有出现之前，有一种叫做电传打字机（Teletype Model 33）的玩意儿，每秒钟可以打10个字符。但是它有一个问题，就是打完一行换行的时候，要用去0.2秒，正好可以打两个字符。要是在这0.2秒里面，又有新的字符传过来，那么这个字符将丢失。于是，研制人员想了个办法解决这个问题，就是在每行后面加两个表示结束的字符。一个叫做“回车”，告诉打字机把打印头定位在左边界；另一个叫做“换行”，告诉打字机把纸向下移一行，这就是“换行”和“回车”的来历。

后来，计算机发明之后，这两个概念也就被般到了计算机上。那时，存储器很贵，一些科学家认为在每行结尾加两个字符太浪费了，加一个就可以。于是，就出现了分歧，一个直接后果是，Windows里的文件在Linux/Mac下打开的话，在每行的结尾可能会多出一个 `^M` 符号。

另一个后果是，由于助教们使用的是Windows和MacOS系统，而OJ平台使用的是Linux系统，文件换行内容的不一致可能导致你的代码测评结果错误，助教们在出题时会尽可能修改文件的换行内容，但你也需要对此保持警惕。

有些同学会发现，示例代码输出时只用了 `\n` 但在Windows环境中也可以正常回车+换行，这是因为C语言在编译代码时，根据所使用的操作系统将 `\n` 自动替换成了操作系统规定的换行内容。

J 科学计数法!

| 难度 | 考点 |

| ---- | ---------- |

| 4 | 字符、字符串 |

题目分析

本题难度稍大一点点，涉及到字符串的读取和字符的相关操作，对新手不做强制要求，下面仅提供解本题的一种思路；

* 读入字符串，根据小数点的位置进行分类，分为指数大于等于0和小于0的情况；

* 若指数大于等于0，根据小数点的位置计算出指数即可；

* 若指数小于0，根据小数点的位置计算出指数，另外还需要根据第一个非0数的位置判断底数部分是整数还是小数；

* 根据题目要求进行输出

另外本题保证输入的是有效数字，且一定是小数，不需要考虑输入数据的前导零或者后置零的情况。

参考代码

```c
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

int main()

{

    char s[200];

    int len, pos = 0;

    scanf("%s", s);

    len = strlen(s); //字符串的长度

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

        if (s[i] == '.')
```

```
        {
            pos = i; //找到小数点的位置
            break;   // break跳出循环
        }
    if (s[0] == '0') //如果这个数小于1，即指数小于0
    {
        int pos2 = 0;
        for (int i = 2; i < len; i++)
            if (s[i] != '0')
            {
                pos2 = i; //找到第一个非0的位置
                break;    // break跳出循环
            }
        putchar(s[pos2]); //输出底数
        if (pos2 != len - 1)
            putchar('.'); //只要第一个非0数的位置不是在最后一位，就输出小数点
        for (int i = pos2 + 1; i < len; i++)
            putchar(s[i]);
        printf("e%d\n", 1 - pos2); //输出剩余部分
    }
    else //如果这个数大于1，即指数大于等于0
    {
        printf("%c.", s[0]); //输出底数和小数点
        for (int i = 1; i < len; i++)
            if (s[i] != '.')
                putchar(s[i]);    //输出小数部分
        printf("e%d\n", pos - 1); //输出指数
    }
}
```