2018级 航空航天类第四次上机赛题解

A 小Z存钱

分析

该题主要考察循环，循环次数即为天数n

代码

```c
#include <stdio.h>

int main()
{
        int n,m,i;
        double rate,ans;
        scanf("%d%d",&n,&m);
        scanf("%lf",&rate);
        ans=m;
        for(i=0;i<n;i++)
        {
                ans*=(1+rate);                  //等同于ans=ans*(1+rate);
        }
        printf("%.2lf",ans);
        return 0;
}
```

B CWD的“完全回文数”

分析

判断回文字符串最简单的方式就是颠倒后判断是否与原来相同

代码

```c
int main()
{
        char s[10];
        scanf("%s", s);
        int flag = 1;
        for (int i = 0; i < 5; i++)
        {
                if (s[i] != s[5 - i - 1])
                {
                        flag = 0;
                        break;
                }
        }
        if (flag)
        {
                printf("1\n");
        }
        else
        {
                printf("0\n");
        }
    return 0;
}
```

C Ausar的数字反转

分析

只需要通过 `%10` 和 `/10` 我们就可以得到输入数字的每一位。再通过 `*10` 和 `+` ，就可以将数字反转过来。

代码

```
#include <stdio.h>

int main()
{
        int a;
        scanf("%d",&a);
        int res = 0;
        while (a > 0)
        {
                res = res * 10 + a % 10;
                a /= 10;
        }
        printf("%d", res);
        return 0;
}
```

D bzb去学院路开会

分析

该题需要从输入的一串字符中分辨出单词的个数，而识别单词的关键在于注意到单词与单词之间是用**空格**隔开的，因此可以循环判断每个字符是否为空格，记录空格的个数，最后通过空格的个数+1得到单词的个数；循环的终止条件为判断字符是否为英文句号

代码

```c
#include <stdio.h>

int main()
{
    int tot = 0;
    char ch;
    while ((ch = getchar()) != EOF)
        {
        if (ch == ' ') tot++;
        if (ch == '.') break;
    }
    printf("%d\n", tot + 1);
    return 0;
}
```

E 好惨一男的

分析

该题需要通过判断输入的成绩与60的大小关系，选择合适的计算公式计算GP，最后代入GPA的计算公式即可得到最终的答案；

需要注意的是，输入的成绩均为**整数**，但最终的GPA为保留两位小数的**实数**，需要人为地在计算公式中加入小数点，将整数的计算转化为实数的计算

代码

```c
#include <stdio.h>

int main()
{
	int a,b,c;
	double ga,gb,gc;

	scanf("%d%d%d",&a,&b,&c);

	if (a<60)
		ga = 0;
	else
		ga = 4 - 3 *(100-a)*(100-a) / 1600.;

	if (b<60)
		gb = 0;
	else
		gb = 4 - 3 *(100-b)*(100-b) / 1600.;

	if (c<60)
		gc = 0;
	else
		gc = 4 - 3 *(100-c)*(100-c) / 1600.;

	printf("%.2lf\n", (5*ga + 4*gb + 2*gc) / 11.);
	return 0;
}
```

可以看出计算不同课程的GP时所使用的方法都是一样的，为了避免重复化的工作，我们可以将实现一个功能的一段程序变成一个单独的模块，也就是所谓的**函数**，每次需要实现该功能时调用这个函数就可以啦！下面的程序就是用函数的方式实现的，是不是简洁很多呢~

```c
#include <stdio.h>
double GP(int x)    //定义函数GP，用来计算不同课程的GP
{
	if (x < 60)
	{
		return 0;
	}
	return 4 - 3.0 * (100 - x) * (100 - x) / 1600;
}

int main()
{
	int a, b, c;
	double ans;
	scanf("%d%d%d", &a, &b, &c);
	printf("%.2f\n", (GP(a) * 5 + GP(b) * 4 + GP(c) * 2) / (5 + 4 + 2));    //调用了
	return 0;
}
```

F 成绩统计1

分析

一共有 $\frac{800}{m}+1$个区间，使用数组记录一下区间的值即可。

代码

```
#include <stdio.h>
int times[805];
int main()
{
        int m;
        scanf("%d", &m);
        int s;
        while (~scanf("%d", &s))
        {
                times[s / m]++;
        }
        for (int i = 0; i*m < 800; i++)
        {
                printf("[%d,%d):%d\n", i*m, (i + 1)*m, times[i]);
        }
        printf("800:%d\n",times[800/m]);
    return 0;
}
```

G Ange的信

分析

该题要对输入的两个日期进行对比，如果是同月同日，则为纪念日，这里隐藏了一个第二行的年份应该大于第一行年份的信息；

其余情况下则需按要求输出落款，这里有两个难点：

- 根据不同的日期选择序数词的表达方式，总共有四种情况，即1、21、31的后缀为st，2、22的后缀为nd，3、23的后缀为rd，其余情况均为th；可用switch中的case, default完成该功能
- 根据不同的月份按照输入的整数mm与相应的英文缩写之间的对应关系进行输出，这里的对应可以采用数组的下标与元素之间的映射关系；设置一个数组（二维数组）按顺序存放月份的缩写字符串，便可以通过检索数组的下标mm检索到对应的月份缩写

代码

```c
#include <stdio.h>

int main()
{
    int y, m, d;
    int yy, mm, dd;
    char month[12][5] = { "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sept

    while(~scanf("%d%d%d%d%d%d", &y, &m, &d,&yy, &mm, &dd))
    {
        if((m == mm) && (d == dd) && (y < yy))        //注意对年份的判断
            printf("Oops!\n");
        else
        {
            switch (dd)
            {
                case 1:
                case 21:
                case 31:
                    printf("%dst ", dd);
                    break;
                case 2:
                case 22:
                    printf("%dnd ", dd);
                    break;
                case 3:
                case 23:
                    printf("%drd ", dd);
                    break;
                default:
                    printf("%dth ", dd);
                    break;
            }
            printf("%s.\n",month[mm-1]);
        }
    }
    return 0;
}
```

H 删除

分析

如果二进制是全0或者全1，删除任意一位即可。

对于剩下的情况，我们假设二进制串为`x`，每一项为x_i（从低位到高位，即从右往左），一共有n项。那么删除第k位之后，剩下的值为：

$$
\begin{aligned}
&\sum_{i=1}^{k-1} x_i \cdot 2^{i-1} + \sum_{i=k+1}^{n} x_i \cdot 2^{i-2} \\
=&\sum_{i=1}^{n} x_i \cdot 2^{i-1} + \sum_{i=k+1}^{n} x_i \cdot 2^{i-2} - x_k \cdot 2^{k-1} - \sum_{i=k+1}^{n} x_i \cdot 2^{i-1} \\
=&\sum_{i=1}^{n} x_i \cdot 2^{i-1} - x_k \cdot 2^{k-1} - \sum_{i=k+1}^{n} x_i \cdot 2^{i-2}
\end{aligned}
$$

其中$\sum_{i=1}^{n} x_i \cdot 2^{i-1}$的和是固定的

所以我们只需要使$x_k \cdot 2^{k-1} + \sum_{i=k+1}^{n} x_i \cdot 2^{i-2}$和最小即可

因为不存在前导0的情况，所以我们假设假设第k位到第n位全是1，不难发现：

$$
1 \cdot 2^{k-1} + \sum_{i=k+1}^{n} 1 \cdot 2^{i-2} = 2^{n-1}
$$

也就是说，我们删除最高位连续1中任意一位对结果是没有影响的。

直到$x_k = 0$时(记此时k为k_1):

$$
ans = 0 \cdot 2^{k-1} + \sum_{i=k+1}^{n} 1 \cdot 2^{i-2} < 2^{n-1}
$$

对于删除任何比最高位0更低的一位：

$$
\begin{aligned}
&x_k \cdot 2^{k-1} + \sum_{i=k+1}^{n} x_i \cdot 2^{i-2} \\
=&x_k \cdot 2^{k-1} + \sum_{i=k+1}^{k_1-1} x_i \cdot 2^{i-2} + ans \\
\leq&ans
\end{aligned}
$$

所以我们可以得出结论，删除最高位0的结果是最大的。

代码

```c
#include <stdio.h>
char s[100005];
int main()
{
	char c;
	int flag = 1, k = 0;
	while ((c=getchar())!=EOF)
	{
		if (c != '0'&&c != '1')
			break;
		if (c == '0' && flag)
		{
			flag = 0;
		}
		else
		{
			s[k++] = c;
		}
	}
	if (flag) k--;
	for (int i = 0; i < k; i++)
	{
		putchar(s[i]);
	}
	return 0;
}
```