

## 第二次上机测验

---

1. 不调用库函数，自行编写函数 mySin(x)，利用公式计算 x 的正弦值，最后一项的绝对值小于  $10^{-6}$  时停止计算，公式为：

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

```
double mySin1(double x)
{
    int b=1, i = 1, sign=1;
    double sum=0, a=x, item = sign*a/b;
    while(fabs(item) > 1e-7)
    {
        sum += item;
        ++i;
        sign = -sign;
        a = pow(x, 2*i-1);
        b = MyFact(2*i-1);
        item = sign*a/b;
    }
    return sum;
}
```

```
int MyFact(int n)
{
    int f = 1;
    for(int i = 2; i <= n; ++i)
        f *= i;
    return f;
}
```

```
int main()
{
    double x;
    scanf("%lf", &x);
    printf("%.6f", mySin1(x));
    return 0;
}
```

---

```
double mySin2(double x)
{
    double sum = x, a =x, item;
    int b = 1, i = 1;
    while(fabs(item) > 1e-7)
    {
        ++i;
        a *= -x*x;
        b *= (2*i-1) * (2*i-2);
        item = a/b;
        sum += item;
    }
    return sum;
}
```

```
int main()
{
    double x;
    scanf("%lf", &x);
    printf("%.6f", mySin2(x));
    return 0;
}
```

算法3减少了计算量;

当 某一项 不太大, 而 分子 或 分母 很大, 以至于超出计算机所能表示的数值范围时, 算法3仍能得出正确的结果 (由于算法3不直接计算 分子 或 分母, 因此算法3可行性更好)。

但算法3会带来精度损失, 因为 每一项 是基于 前一项 的计算结果的, 所以精度损失会不断叠加。

```
double mySin3(double x)
{
    double sum, item, i=1;
    sum = item = x;
    while(item > 1e-7 || item < -1e-7)
    {
        ++i;
        item *= (- x*x) / ((2*i-1)*(2*i-2));
        sum += item;
    }
    return sum;
}
```

```
int main()
{
    double x;
    scanf("%lf", &x);
    printf("%.6f", mySin3(x));
    return 0;
}
```

- 
2. 11.11 全球购物节就要到了，某护肤品商家搞活动——五个空瓶子可以换一瓶同款护肤水。假定每瓶护肤水卖  $p$  元，小花准备花  $m$  元钱。编程求解她最多可以买到多少瓶护肤水。

```
int Shopping(int p, int m)
{
    int nn, n = m/p;
    int sum = n;           //本来可以买 n 瓶护肤水, n 个空瓶子
    while( (n/5) != 0)
    {
        count = n/5;       //可以换 count 瓶护肤水
        sum += count;
        n = n%5 + count;   //又得 n 个空瓶子
    }
    return sum;
}
```

```
int main()
{
    int price, money;
    cin >> price >> money;
    cout << Shopping(price, money);
    return 0;
}
```

```
int change(int n);
```

```
int count = 0;
```

---

```
int main()
```

```
{    int price, money;  
    cin >> price >> money;
```

```
    int n = money/price ;
```

```
    cout << n + change(n);
```

```
    return 0;
```

```
}
```

```
int change(int n)
```

```
{    if(n < 5)
```

```
        return count;
```

```
    else
```

```
{    count += n/5;
```

```
        return change(n/5 + n%5);
```

```
    }
```

```
}
```

//本来可以买 n 瓶护肤水, n 个空瓶子  
//可以换 count 瓶护肤水

static int count = 0;



---

```
int myRegain(int) ;
int main()
{
    int p, m;
    cin >> p >> m;
    cout << myRegain(m/p) + m/p << endl;
    return 0;
}
int myRegain(int n)
{
    int h = n / 5;
    if(h)
        return myRegain(n - 5*h + h) + h;
    else
        return 0;
}
```



# 作业5

自检

Ex1. 设计程序，计算下面表达式的值： $1 - 1/2 + 1/3 - 1/4 + \dots + 1/99 - 1/100$ 。

```
double f()  
{  
    int sign = 1;  
    double sum = 1.0;  
    for(int n = 2; n <= 100; ++n)  
    {  
        sign = -sign;  
        sum += sign * 1.0/n ;  
    }  
    return sum;  
}
```

计次型循环

- Ex2. 所谓完数，指的是一个特殊的整数，它等于其所有因子（除自身之外）之和，例如 $6=1+2+3$ 。设计程序，输出  $[1, n]$  之间的所有完数， $n$  为正整数。

10  
6

100  
6 28

6 28

```
int Perfect(int i) //bool
{
    int perfectNumber = 0;
    for(int j = 1; j < i; ++j) //sqrt(i)+1, i/2
        if(i % j == 0)
            perfectNumber += j;
    if(i == perfectNumber) return 1; //true
    else return 0; //false
}
```

```
void Display(int n)
{
    for(int i = 1; i <= n; ++i)
        if(Perfect(i))
            cout << i << endl;
}
```

Ex3. 设计程序，识别三个不同数中第二大的数（假定均为正整数）。

```
int main()
{
    int i, j, k;
    //for(int i=1; i <= 6; ++i)
    //{
        scanf("%d%d%d", &i, &j, &k);
        printf("%d \n", iMiddle(i, j, k));
    //} //6种排列情况
    return 0;
}
```

方法一:

```
if (i < j)
    if (j > k)
        return j;
    else if (i < k)
        return i;
    else
        return k;
else
    if (j < k)
        return j;
    else if (i > k)
        return i;
    else
        return k;
```

```
if (i < j)
    if (j < k) return j;
    if (k < j) return k;
else
    if (i < k) return i;
    if (k < i) return k;
```



```
(i>j ? (j>k ? j : (i<k?i:k)) : (j<k ? j : (i>k?i:k))) ;
```

方法二:

```
int iMiddle(int i, int j, int k)
```

```
{
```

```
    if((i>j && j>k) || (i<j && j<k))
```

```
        return j;
```

```
    else if((i>j && i<k) || (i<j && i>k))
```

```
        return i;
```

```
    else
```

```
        return k;
```

```
} //求中间数
```

```
if(i<j<k) return j;
```

```
else if(j<i<k) return i;
```

```
else return k;
```

或

```
if((i<j<k) || (k<j<i)) return j;
```

```
else if((j<i<k) || (k<i<j)) return i;
```


```
else return k;
```



方法三:

```
if ((i-j) * (j-k) > 0)
    return j;
else if ((j-k) * (k-i) > 0)
    return k;
else
    return i;
```

```
if ((x-y) * (y-z) > 0);
    return y;
else if ((y-z) * (z-x) > 0);
    return z;
else
    return x;
```



方法四:

**sum-min-max**

# 第9周自主训练任务

1. 在多模块中分别编写迭代法的函数和递归函数，求埃尔米特（Hermite）多项式中第 $n+1$ 项 $H_n(x)$ 的值，并在main函数中调用、输出结果。 $H_n(x)$ 定义为：

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x) \quad (n > 1)$$

（测试数据：输入0 3，输出1；输入1 3，输出6；输入2 3，输出34；输入5 3，输出3816）

（可尝试由两人或三人合作完成，并交换角色分别体会头文件的作用）

---

```
double Hermite_Iterative(int n, double x)
```

```
{    if(n == 0)
        return 1;
    else if(n == 1)
        return 2*x;
```

```
else
```

```
{    double res1 = 1, res2 = 2*x;
    double Result = 0;
    for(int i = 2; i <= n; ++i)
    {    Result = 2*x*res2 - 2*(i-1)*res1;
        res1 = res2;
        res2 = Result;
    }
    return Result;
}
```

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x)$$



---

2. 自行总结变量有哪些属性及其分类。（略）

3. 编程输出N以内的全部同构数（同构数是指一个正整数恰好出现在其平方数的最右端，如， $376*376 = 141376$ ）。

```

.....
#define N 10000
int main()
{
    for(int i = 1; i < N; ++i)
        if(Isomorph(i))
            cout << i << ' ';

    return 0;
}

```

```

.....
int main()
{
    int n;
    cin >> n;
    for(int i = 1; i < n; ++i)
        if(Isomorph(i))
            cout << i << ' ';

    return 0;
}

```

```

bool Isomorph(int i)
{
    if(i < 10 && i == i*i % 10)
        return true;
    else if(i < 100 && i == i*i % 100)
        return true;
    else if(i < 1000 && i == i*i % 1000)
        return true;

    .....
    return false;
}

```

```
.....
int main()
{
    int m = 10;
    for(int i = 1; i < N; ++i)
    {
        if(i == m)
            m *= 10;
        if(IsomorphM(i, m))
            cout << i << ' ';
    }
    return 0;
}
```

```
bool IsomorphM(int i, int m)
{
    if(i == i*i % m)
        return true;
    else
        return false
}
```

// m 随 i 的位数增大而倍增

- 

不是三角形

# 等边

## 等腰非等边非直角

## 直角非等腰

## //输入的边长有可能形成等腰直角三角形吗？

```

if (a+b<=c || a+c<=b || b+c<=a)
    return 0;
else if (a==b && b==c)
    return 1;
else if (a==b || b==c || c==a)
    return 2;
else if (a*a+b*b==c*c || a*a+c*c==b*b || b*b+c*c==a*a)
    return 3;
else
    return 4;

```

```
cout << "a = " << b << ", b = " << a << endl; // 囧
```

## 5. 编程实现：交换两个int型变量的值，不引入第三个变量。

```
int a = 5, b = 9;  
a = a + b;  
b = a - b;  
a = a - b;  
cout << a << b;
```

如果a、b不是5、9，要注意溢出问题，即在a、b之和（差）溢出时，该方法不能奏效。

```
int a = 5, b = 9;  
a = a ^ b;  
b = a ^ b;  
a = a ^ b;  
cout << a << b;
```

a 0000 0101  
b 0000 1001  
a 0000 1100  
b 0000 1001  
b 0000 0101  
a 0000 1001

$(a \oplus b) \oplus c$  为  $a \oplus (b \oplus c)$   
 $a \oplus a$  为 0  
0  $\oplus a$  为 a  
 $a \oplus b$  为  $b \oplus a$

- 
6. (选做) 五猴分桃：五只猴子采了一堆桃子，它们约定次日早晨起来再分。半夜里，一只猴子偷偷起来把桃子平均分成五堆后，发现还多一个，于是吃了这个桃子，拿走了其中一堆；第二只猴子醒来，又把桃子均分成五堆后，还是多了一个，它也吃了这个桃子，拿走了其中一堆；第三、第四、第五只猴子都依次如此做了。设计程序，求原先这堆桃子至少有多少个？最后剩多少个桃子？（已知 int 范围内有解：3121，1020）

# 思路

第1只猴子

从6开始穷举

```
int monkey=1, amount=6, peach=amount;    //peach为分前桃子数

while(monkey <= 5)
    if(peach%5==1 && peach>5) //可以继续分, 保持迭代
    {
        peach = 4*(peach-1)/5;
        ++monkey;
    }
    else //不能分, 穷举下一个可能的值, 重新迭代(回溯)
    {
        amount += 5 ;
        peach = amount;
        monkey = 1;
    }
cout << amount << ", " << peach << endl;    //第5只猴子也能分
```



# 递归

```
cout << PeachR(amount, 1);
```

```
int amount = 6;      //全局变量
int PeachR(int peach, int monkey)
{
    if(monkey > 5)
        return amount;
    else if(peach%5==1 && peach>5) //可以继续分
        return PeachR(4*(peach-1)/5, ++monkey);
    else //不能分, 重新再来
        amount += 5;
        return PeachR(amount, 1);
}
```

# 用符号常量改进

```
#define N 5
int amount = N + 1;           //全局变量
```

```
cout << PeachR(amount, 1);
```

```
int PeachR(int peach, int monkey)
{
    if(monkey > N)
        return amount;
    else if(peach%N ==1 && peach>N) //可以继续分
        return PeachR( (N-1) * (peach-1) /N, ++monkey);
    else //不能分, 重新再来
        amount = amount + N;
        return PeachR(amount, 1);
}
```

```
int p = 1,n;  
for(p;;++p)  
{
```

同学①

自检

```
    n = p;  
    if(n % 5 == 1)  
    {  
        n = (n-1)*4/5;  
        if(n % 5 == 1)  
        {  
            n = (n-1)*4/5;  
            if(n % 5 == 1)  
            {  
                n = (n-1)*4/5;  
                if(n % 5 == 1)  
                {  
                    n = (n-1)*4/5;  
                    cout << p << ", " << n;  
                    return 0;  
                }  
            }  
        }  
    }  
}
```

容易理解

重复代码

如果不止5只猴子呢？

## 对前一种做法的改进

用了半结构化语句

```
int p = 1, n, i;
for(p; ; ++p)
{
    n = p;    //桃子总数保留
    for(i = 1; i <= 5 && n % 5 == 1; ++i)
        n = (n - 1) * 4 / 5;
    if(i > 5)
        break; //说明内循环循环了五次才结束
}
cout << ...
```

不建议这种基于数据类型的做法，  
不利于计算思维训练

```
int sum;
double n = 1.1, p;
p = n;
for(sum = 1 ; p != int(n) ; ++sum)
{
    n = sum;
    for(int i = 1; i <= 5; ++i)
    {
        n = (n - 1) * 4 / 5;
    }
    p = n;
}
--sum;
cout << sum << ", " << n;
```

```
int peaches = 6;
int peach(int i, int n)  // i为桃子数, n为猴子数
{
    if (n >= 1)          //没分完
    {
        if (i % 5 != 1)
        {
            peaches += 5;
            i = peaches;
            n = 5;
        } // 回溯
        peach((i - 1) / 5 * 4, n - 1) ; // 分桃
    }
    return peaches;
} //cout << peach(6, 5) <<endl;
```

这种递归形式不是所有的编译器能通过，  
即使通过，有可能出错，  
每个分支的返回值不明确

? !

```
#include<iostream>
using namespace std;
int main()
{
    cout<<"3121 1020";
    return 0;
}
```

# Thanks!

---

