

# 《程序设计基础》实验一报告

---

作者：李文骏

学号：2020213969

日期：2021-10-19

## 1.实验目的要求

---

- (1) 使用C++语言编写简单的计算机程序，掌握C++程序从编辑、编译到运行的全过程，初步了解C++程序的特点。
- (2) 掌握C++语言的数据类型（包括对各种数据类型变量的定义、初始化、赋值等）、运算符和表达式的使用。
- (3) 掌握数据的输入输出方法。
- (4) 理解基本的解题技巧，掌握用自顶向下、逐步求精的过程设计算法。
- (5) 熟练掌握if/else语句和switch语句的使用。
- (6) 熟练掌握while语句、do/while语句和for语句的使用以及区别。
- (7) 掌握break和continue程序控制语句的使用。
- (8) 理解并掌握如何模块化的用函数来构建程序。
- (9) 掌握定义函数的方法，了解函数的重载方法，了解内联函数的机制。
- (10) 掌握函数间传递信息的机制。
- (11) 掌握函数的嵌套调用（和递归调用）的方法。

## 2.实验设备与环境

---

硬件环境：

- Huawei Matebook 14锐龙版笔记本一台
- PU为AMD Ryzen 7 4800U with Radeon Graphics 1.80 GHz，内存为16GB，硬盘大小为463GB

软件环境：

- 操作系统：Windows10家庭中文版
- 开发工具：Visual Studio Code&Visual Studio 2019(两者都有用到)

## 3.实验内容

---

### 第一部分 代码测试与分析

- (1) 阅读下列程序，写出（由指定的输入）所产生的运行结果，并指出其功能。

<1>

功能分析：该程序是将输入的大写字母转换成小写字母，输入的小写字母转换为大写字母

```
#include <iostream>
using namespace std;

int main()
{
    char ch;
    cin >> ch ;
    ch = ( ch >= 'A' && ch <= 'Z' ) ? ( ch + 32 ) : ch ;
    ch = ( ch >= 'a' && ch <= 'z' ) ? ( ch - 32 ) : ch ;
    cout << ch << endl;
    return 0;
}
```

### 代码测试:

输入1:

A

输出1:

A

输入2:

b

输出2:

B

输入3:

8

输出3:

8

对上述输入输出结果进行分析:

首先我们观察程序容易知道, 程序的主干部分是这两行:

```
ch = ( ch >= 'A' && ch <= 'Z' ) ? ( ch + 32 ) : ch ;
ch = ( ch >= 'a' && ch <= 'z' ) ? ( ch - 32 ) : ch ;
```

可当我们写入输入1时, 程序得到输出1是一个意想不到的结果, 大写字母A并没有转化成我们预期的小写字母a, 于是我将第二行代码注释掉, 输出又发生了变化:

```
ch = ( ch >= 'A' && ch <= 'Z' ) ? ( ch + 32 ) : ch ;
//ch = ( ch >= 'a' && ch <= 'z' ) ? ( ch - 32 ) : ch ;
```

此时我们再次输入A:

A

得到输出：

a

算法分析：

由于程序的第一行将大写字母转化成了小写字母，因此得到的小写字母接着执行第二行程序，又变成了大写字母

解决方案：

可通过if-else条件判断对程序进行优化，程序主干代码如下：

```
if (ch >= 'A' && ch <= 'Z')
{
    ch += 32;
}
else if (ch >= 'a' && ch <= 'z')
{
    ch -= 32;
}
```

算法分析：

该进算法通过if和else if进行控制选择，从而使两个转换的操作分条件进行，互不干扰，互不纠缠，选择结构清晰分明。

此时我们就能在符合条件的任何输出下得到我们想要的输出

思考：

当我们遇到结构比较复杂，比较多元的条件时，应当使用if，else-if，while，switch case等更加高级一些的选择结构，避免使用三目条件运算符

---

## <2>

功能分析：判断输入的整数是否大于1000，判断输入的浮点数是否小于1e3(10的3次方)，并且返回判断结果的布尔值

```
#include <iostream>
using namespace std;

int main ()
{
    int m;
    float x;
    bool bi,br;

    cout << "\n int m=";
    cin >> m;
    bi = m > 1000;
    cout << "\n float x=";
    cin >> x;
    br = x <= 1e3;
    cout << bi << ',' << br << endl;
```

```
    return 0;  
}
```

### 代码测试:

输入1:

100 40

输出1:

0,1

---

输入2:

2000 3000

输出2:

1,0

---

输入3:

1000 1000

输出3:

0,1

---

输入4:

2000 300

输出4:

1,1

---

输入5:

100 4000

输出5:

0,0

总结：在这个案例中，我们学到了c++中科学计数法的表示“X e Y”，也就是表示x乘以10的Y次方

---

### <3>

功能分析：用于检验和探索前置与后置连加加减符号对变量的影响

```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cin >> n ;
    if ( n ++ < 10 )
        cout << n << endl ;
    else
        cout << n -- << endl ;
    return 0;
}
```

#### 代码测试:

输入1:

8

输出1:

9

n=9

输入2:

9

输出2:

10

n=10

输入3:

10

输出3:

11

n=10

输入4:

11

输出4:

12

n=11

总结:

该程序的陷阱代码主要是:

```
if ( n ++ < 10 )
```

```
else  
    cout << n -- << endl ;
```

括号内的语句实质上是:

```
n<10;n+=1;
```

else语句内n--实质上是:

```
cout<<n;n-=1;
```

运算符前置, 则先执行该变量的递增(减)操作, 然后再执行该变量的其他的操作

运算符后置, 则先执行该变量的相关操作, 再执行该变量的递增(减)操作

## <4>

功能分析: 该程序是运用**辗转相减法**求1000和850的最大公因数

```
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int m,n;  
    m = 1000;  
    n = 850;  
    cout << "\n(" << m << ', ' << n << ')';  
    while ( m != n ) {  
        while ( m > n ) {  
            m = m - n ;  
            cout << '(' << m << ', ' << n << ')';  
        }  
        while ( m < n ) {  
            n = n - m ;  
            cout << '(' << m << ', ' << n << ')';  
        }  
    }  
    cout << "\n" << m ;  
    return 0;  
}
```

输出为：

```
(1000,850)(150,850)(150,700)(150,550)(150,400)(150,250)(150,100)(50,100)(50,50)
50
```

通过查阅资料以及自主设计，了解到求最大公因数的方法还有穷举法、辗转相除法、更相减损法等

**穷举法：**

```
//求最大公约数方法1：穷举法
private static int maxCommon1(int a,int b){
    if(a<b){ //如果a<b，先把ab交换，方便以后操作
        int tmp=a;
        a=b;
        b=tmp;
    }
    //如果b能直接整除a，说明b是它们的最大公约数
    if(a%b==0){
        return b;
    }
    //否则从小的开始依次整除，当a和b同时能整除那个数的时候说明就是它们的最大公约数
    for(int i=b-1;i>1;i--){
        if(a%i==0&&b%i==0){
            return i;
        }
    }
    return 1; //说明除完还没有找到，只能返回1
}
```

**辗转相除法：**

```
//求最大公约数方法3：辗转相除法

/**
 * 用大数对小数求余，若余数为0，则除数为最大公约数。
 * 若余数不为0，将此余数作为除数，小数作为被除数，重新求余，直到余数为0为止。
 * 此时的最大公约数为余数。例如：27和6。 27%6=3,6%3=0.所以最大公约数为3.
 */
private static int maxCommon3(int a,int b){
    if(a<b){ //如果a<b，先把ab交换，方便以后操作
        int tmp=a;
        a=b;
        b=tmp;
    }
    int n=a%b;
    while(a%b!=0){
        a=b;
        b=n;
        n=a%b;
    }
    return b;
}
}
```

**更相减损法：**

```

//更相减损法求最大公约数
int gcd1(int a, int b) {
    int i = 0, temp, x = 0;
    while (a % 2 == 0 && b % 2 == 0) {    //m,n有公约数2时
        a /= 2;
        b /= 2;
        i += 1;
    }
    //a,b的值互换
    if (a < b) {
        temp = a;
        a = b;
        b = temp;
    }
    while (x) {
        x = a - b;
        a = (b > x) ? b : x;
        b = (b < x) ? b : x;
        if (b == (a - b)) {    //差和减数相等
            break;
        }
    }
    if (i == 0) {
        return b;
    }
    else {
        return (int)pow(2, i)*b;
    }
}

```

## <5>

功能分析：将两数相除，并返回商(k)和余数(m)

```

#include <iostream>
using namespace std;

int main()
{
    int m,n,k ;

    m = 1000 ;
    n = 45 ;
    cout << "\n(" << m << ':' << n << ')' ' ' ;
    k=0;
    while ( m >=n ){
        m = m - n ;
        k = k + 1 ;
    }
    cout << k << "----" << m << endl ;

    return 0;
}

```

输出：



思考题：

上述程序是对最初的m和n做除法运算，最后输出的k是商，m是余数

---

## <6>

功能分析：对于1-5这五个数，是奇数则输出\*#，是偶数则不输出

```
#include <iostream>
using namespace std;

int main()
{   int i;
    for ( i = 1 ; i <= 5 ; i ++ ){
        if ( i % 2 )
            cout << '*';
        else
            continue;
        cout << '#';
    }
    cout << "$\n" ;

    return 0;
}
```

输出：

```
*#*#*#*$
```

分析：

该程序存在一个容易让人迷惑的地方，首先应明确continue语句一旦执行，则不会再运行continue以下的代码，而是重开一次循环，所以如果是偶数的话，是不会输出#的，而是没有任何输出，直接开始下一循环，若为奇数的话，continue语句不会执行，则\*和#都会输出

---

## <7>

功能分析：该程序主要是考察了do-while结构中无论是否符合while中的条件，程序都会执行一次内容，然后再判断是否要继续或是退出

```
#include <iostream>
using namespace std;

int main()
{   int a = 1,b = 10;

    do{
        b -= a ;
        a ++ ;
    }while ( b -- <= 0 ) ;
    cout << "a=" << a << " ,  b=" << b <<endl ;

    return 0;
}
```

输出:

a=2,b=8

思考题解答:

上述程序a的值为2, b的值为8, 循环进行了1次, 如果改为while循环, 循环进行0次

## <8>

功能分析: 该程序的两个函数分别实现了求两个数的最大公因数和最小公倍数的功能, 并返回这两个最大公因数和最小公倍数

且求最大公因数时用的是欧几里得除法

```
#include <iostream>
using namespace std;

int hcf( int u , int v )
{
    int a,b,t,r;
    if (u > v) {
        t = u ;
        u = v ;
        v = t ;
    }
    a = u ;
    b = v ;
    while ( ( r = b % a ) != 0 ){
        b = a ;
        a = r ;
    }
    return a;
}

int lcd( int u , int v , int h )
{
    return (u * v / h) ;
}

int main()
{
    int u , v , h , l ;

    cin >> u >> v ;
    h = hcf ( u , v ) ;
    cout << "hcf=" << h << endl ;
    l = lcd ( u , v , h ) ;
    cout << "lcd=" << l << endl ;
    return 0;
}
```

输入:

120 500

输出：

```
hcf=20
lcd=3000
```

思考题解答：

函数int hcf( int u , int v )实现的是求u和v这两个数的最大公因数的功能，函数int lcd( int u , int v , int h )实现的是求这两个数最小公倍数的功能。函数的形参含义是指函数与变量之间的一种沟通媒介，将变量的值传递给参数，但是形参的值在函数体内进行运算的时候，并不影响函数外实参的值。

---

## <9>

功能分析：利用函数递归求解斐波那契数列前n项

```
#include <iostream>
using namespace std;

long fib ( int g )
{   switch ( g ){
        case 0 : return 0;
        case 1 :
        case 2 : return 1;
    }
    return ( fib( g - 1 ) + fib( g - 2 ) ) ;
}

int main()
{   for(int i = 1;i<=20;i++){
        cout<<fib(i)<<" ";
    }
    return 0;
}
```

输出：

```
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
```

思考题解答：

递归函数long fib ( int g )的功能是将斐波那契数列的前n项求出来，其具体过程分为两部，递推加回归，先从long fib(n)递推到long fib(1)和long fib(0)这两个已知值，然后再进行回归倒推出long fib(n)的值

利用递归的思想我们可以将很多多层次的复杂问题简单化，例如大多数迭代式和或者乘积等

---

## <10>

功能分析：该程序很好地体现了变量的作用域这个知识点

```
#include <iostream>
using namespace std;
```

```

int k = 1;

void fun( int m )
{
    m += k;
    k += m;
    {
        char k = 'B';
        cout << "(2)" << char(k-1) << endl ;
    }
    cout << "(3)" << m << ',' << k << endl ;
}

int main()
{
    int i = 4 ;
    int fun (int);
    fun( i ) ;
    cout << "(1)" << i << ',' << k << endl ;
    return 0;
}

```

输出：

```

(2)A
(3)5,6
(1)4,6

```

思考题解答：

在全局域中，k为int型，值为1，在fun之中，代码块之外，k为int型，代码块中，k被重定义为char型，在main函数中，k为int型

## <11>

```

#include <iostream>
#include<iomanip>
using namespace std;

void subp ( )
{
    static int x = 0 , y = 0 ; //问题 (1)
    int a = 1, b = 1;
    a = a + x ;
    b = b + y ;
    cout<<"subp函数输出: \n";
    cout << setw(5) << a << setw(5) << b << '\n' ;
    cout << setw(5) << x << setw(5) << y << '\n' ;
    x++;
    y++;
}

int x,y;

int main()
{
    int a = 9, b = 3;
    x = a - b ;
}

```

```

    y = a + b ;
    subp();
    cout<<"main函数输出: \n";
    cout << setw(5) << a << setw(5) << b << '\n' ;
    cout << setw(5) << x << setw(5) << y << '\n' ;
    subp();
    return 0;
}

```

输出：

```

subp函数输出:
    1    1
    0    0
main函数输出:
    9    3
    6   12
subp函数输出:
    2    2
    1    1

```

思考题解答：

由于静态变量在计算机内存中地址不变，不会因为函数外的操作而改变，也不会因为重定义而改变值（也即static后面的初始化语句只会执行一次）

同时该程序还使用了iomanip库中的函数setw()，是为了输出空格

## 第二部分 编写程序实现下列问题的求解

### 1.用cmath实现

1.3

$$\sin X + \cos X + \arctan X$$

```

#include<iostream>
#include<cmath>
using namespace std;
const double PI = acos(-1);
int main(){
    double x;
    cin>>x;
    cout<<sin(x)+cos(x)+atan(x);
    return 0;
}

```

1.4

$$e^{x+y} + e^{x-y}$$

```
#include<iostream>
using namespace std;
int main()
{
    double x,y;
    cin>>x>>y;
    cout<<exp(x+y)+exp(x-y);
    return 0;
}
```

## 2.数字反转输出

对任意输入的四位整数，分别求出其各位数字，并按从后到前的顺序依次输出。例如，输入为1234时，输出结果为4，3，2，1。

```
#include<iostream>
using namespace std;

int main()
{
    int a,b[4]{};
    cin>>a;
    for(int i=0;i<4;i++)
    {
        b[i] = a%10;
        a/=10;
    }
    for(int i=0;i<3;i++)
    {
        cout<<b[i]<<",";
    }
    cout<<b[3]<<endl;
}
```

输入：

1234

输出：

4,3,2,1

## 3.求解下面函数值

<3><4>选做一题

选择<3>

$$z = e^{x+y} (x < 0, y < 0)$$

$$z = \ln(x+y) (1 \leq x+y \leq 10)$$

$$z = \lg|x+y| + 1, \text{ else}$$

```
#include<iostream>
#include<cmath>
```

```

using namespace std;
int main()
{
    double x,y;
    cin>>x>>y;
    if(x<0&&y<0)
    {
        cout<<exp(x+y);
    }
    else if(x+y>=1&&x+y<=10)
    {
        cout<<log(x+y)
    }
    else
    {
        cout<<log10(fab(x+y)+1);
    }
    return 0;
}

```

#### 4.编程求解下列各计算式

由题：选择（2）（3）两题

2)

```

#include<iostream>
using namespace std;
long function(int n)
{
    long sum = 1;
    for (int i = 1; i < n + 1; i++)
    {
        sum *= i;
    }
    return sum;
}
int main()
{
    long sum = 1;
    for (int i = 1; i < 8; i++)
    {
        sum += function(i);
    }
    cout << sum;

    return 0;
}

```

输出结果为：5914

3)

该题我们可以利用数列的思想：

通项公式为：

$$a_n = n * (n + 1)$$

而题目则是求数列的前n项和

```
#include<iostream>
using namespace std;
int main()
{
    int sum = 0;
    for(int i=1;i<40;i+=2)
    {
        sum+=i*(i+1);
    }
    cout<<sum;
    return 0;
}
```

输出结果为：

11060

## 5.打印下面图形

```
#include<iostream>
using namespace std;
int main()
{
    for(int i=0;i<=10;i++)
    {
        for(int j=0;j<10-i;j++)
        {
            cout<<" ";
        }
        for(int a=0;a<=i;a++)
        {
            cout<<2*a+1<<' ';
        }
        int k = i;
        while(k>=0)
        {
            k--;
            cout<<2*k+1<<" ";
        }
        cout<<endl;
    }
}
```

输出结果如下：



```
Microsoft Visual Studio 调试控制台

      1
     1 3 1
    1 3 5 3 1
   1 3 5 7 5 3 1
  1 3 5 7 9 7 5 3 1
 1 3 5 7 9 11 9 7 5 3 1
1 3 5 7 9 11 13 11 9 7 5 3 1
1 3 5 7 9 11 13 15 13 11 9 7 5 3 1
1 3 5 7 9 11 13 15 17 15 13 11 9 7 5 3 1
1 3 5 7 9 11 13 15 17 19 17 15 13 11 9 7 5 3 1
1 3 5 7 9 11 13 15 17 19 21 19 17 15 13 11 9 7 5 3 1

C:\Users\Lee\source\repos\Project1\Debug\Project1.exe (进程 19868) 已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。...
```

算法设计思路说明：

1. 每一行的左侧先生成对应每行数量的空格
2. 在左侧接着每一行的空格之后按照 $2n+1$ 的通项生成数字一直到中间最大项
3. 按照 $2n+1$ 的公式 $n$ 逐渐递减生成数字
4. 最后在右侧补上空格
5. 循环每一行直至中间最大项为21

## 6.编程产出数对<i,j>

编程产生出1到10以内的所有数对<i,j>并输出,其中 $i>j$ 。

```
#include<iostream>
using namespace std;
int main()
{
    for(int i=1;i<=10;i++)
    {
        for(int j=1;j<i;j++)
        {
            cout << '<' << i << ',' << j << '>' << ' ';
        }
    }
    return 0;
}
```

输出：

```
<2,1> <3,1> <3,2> <4,1> <4,2> <4,3> <5,1> <5,2> <5,3> <5,4> <6,1> <6,2> <6,3>
<6,4> <6,5> <7,1> <7,2> <7,3> <7,4> <7,5> <7,6> <8,1> <8,2> <8,3> <8,4> <8,5>
<8,6> <8,7> <9,1> <9,2> <9,3> <9,4> <9,5> <9,6> <9,7> <9,8> <10,1> <10,2> <10,3>
<10,4> <10,5> <10,6> <10,7> <10,8> <10,9>
```

算法分析：

起初这一行代码 `for(int j=1;j<i;j++)` 我是写成了 `for (int j =1;j<=10;j++)` ,但由于题目中只需要我们生成 $i>j$ 的情况，故后者在资源上有些浪费，从而拖慢了程序运行的速度，因此我改成了前者。

## 7. 求出10000以内符合条件的数

编程求出10000以内的所有符合如下条件的数：其高位数字小于低位数字。如12, 238, 3578等。但21, 548不符合条件。

```
#include<iostream>
using namespace std;
bool isNumber(int n){
    int t = n % 10;
    n /= 10;
    while (n > 0){
        if (t <= n % 10){
            return false;
        }
        t = n % 10;
        n /= 10;
    }
    return true;
}
int main()
{
    for (int i = 1; i < 10000; i++)
    {
        if(isNumber(i))
        {
            cout << i << " ";
        }
    }
    return 0;
}
```

输出：

```
1 2 3 4 5 6 7 8 9 12 13 14 15 16 17 18 19 23 24 25 26 27 28 29 34 35 36 37 38 39
45 46 47 48 49 56 57 58 59 67 68 69 78 79 89 123 124 125 126 127 128 129 134 135
136 137 138 139 145 146 147 148 149 156 157 158 159 167 168 169 178 179 189 234
235 236 237 238 239 245 246 247 248 249 256 257 258 259 267 268 269 278 279 289
345 346 347 348 349 356 357 358 359 367 368 369 378 379 389 456 457 458 459 467
468 469 478 479 489 567 568 569 578 579 589 678 679 689 789 1234 1235 1236 1237
1238 1239 1245 1246 1247 1248 1249 1256 1257 1258 1259 1267 1268 1269 1278 1279
1289 1345 1346 1347 1348 1349 1356 1357 1358 1359 1367 1368 1369 1378 1379 1389
1456 1457 1458 1459 1467 1468 1469 1478 1479 1489 1567 1568 1569 1578 1579 1589
1678 1679 1689 1789 2345 2346 2347 2348 2349 2356 2357 2358 2359 2367 2368 2369
2378 2379 2389 2456 2457 2458 2459 2467 2468 2469 2478 2479 2489 2567 2568 2569
2578 2579 2589 2678 2679 2689 2789 3456 3457 3458 3459 3467 3468 3469 3478 3479
3489 3567 3568 3569 3578 3579 3589 3678 3679 3689 3789 4567 4568 4569 4578 4579
4589 4678 4679 4689 4789 5678 5679 5689 5789 6789
```

## 8. 求任一整数N的标准分解式

求任一整数N的标准分解式，即素数因子之积。例如 $16=2 \times 2 \times 2 \times 2$ ， $15=3 \times 5$

```
#include<iostream>
#include<cmath>
using namespace std;
```

```

int main()
{
    int n;
    cin>>n;
    int i=2;
    while(n/i!=0)
    {
        if(n%i==0)
        {
            n/=i;
            cout<<i<<" ";
            i = 2;//因数重置
        }
        else
        {
            i++;
        }
    }
}

```

输入1:

16

输出1:

2 2 2 2

输入2:

15

输出2:

3 5

## 第三部分 分析与思考

### 1.升降最大子序列

编程求出数列的所有升或降的最大子序列。如下面数列的解为如下:

1,20,30,12,3,5,7,4,6,100,11,8

(1,20,30),(30,12,3),(3,5,7),(7,4),(4,6,100),(100,11,8)。

```

#include <iostream>
#include <vector>
using namespace std;

int main(int argc, char** argv) {
    cout<<"----- 逐个输入数组元素 ， 以空格隔开 ----- "<<endl;
    vector<int> a;
    int i = 0;
    do{
        cin >> i;
        a.push_back(i);
    }while(getchar() !='\n');
}

```

```

int start = 0;
int end;
for(int i = 1; i < a.size()-1; i++){
    if( ((a[i-1]>a[i]) & (a[i]<a[i+1])) | ((a[i-1]<a[i]) & (a[i]>a[i+1])) )
){
        end = i;
        cout<<"(";
        for(int j=start; j<end+1;j++){
            cout<<a[j];
            if(j!=end) cout<<", ";
        }
        cout<<")"<<endl;
        start = end;
    }
    if(i==a.size()-2){
        end = a.size()-1;
        cout<<"(";
        for(int j=start; j<end+1;j++){
            cout<<a[j];
            if(j!=end) cout<<", ";
        }
        cout<<")"<<endl;
    }
}
return 0;
}

```

输入:

1 20 12 3 5 8 4 7 100 12 9

输出:

```

选择Microsoft Visual Studio 调试控制台
----- 逐个输入数组元素 , 以空格隔开 -----
1 20 12 3 5 8 4 7 100 12 9
(1, 20)
(20, 12, 3)
(3, 5, 8)
(8, 4)
(4, 7, 100)
(100, 12, 9)

C:\Users\Lee\source\repos\Project1\Debug\Project1.exe (进程 24300) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口。

```

## 2.编程求12100的末三位数。

编程求12100的末三位数。

```

#include<stdio.h>
int main()
{
    int i,x,y,z=1;
    scanf("%d%d",&x,&y); //输入底数和幂
    for(i=1;i<=y;i++)

```

```

        z = z*x%1000;
    if(z>=100)    //输出结果
        printf("%d",z);
    else
        printf("%d",z);

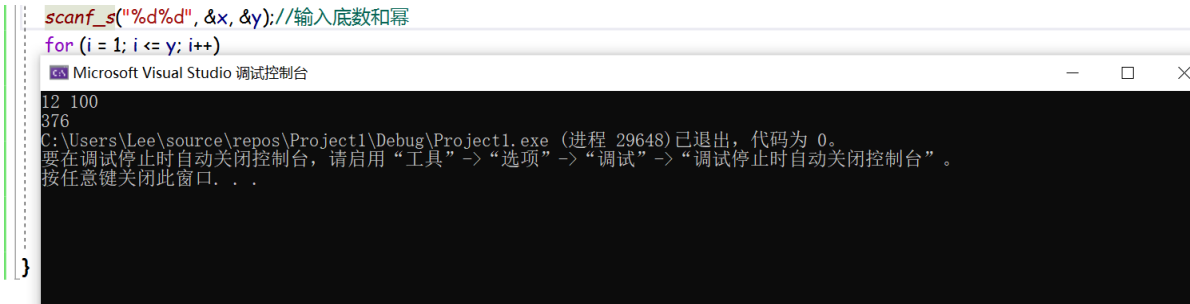
}

```

输入：

12 100

输出：



```

scanf_s("%d%d", &x, &y); //输入底数和幂
for (i = 1; i <= y; i++)
{
    Microsoft Visual Studio 调试控制台
    12 100
    376
    C:\Users\Lee\source\repos\Project1\Debug\Project1.exe (进程 29648) 已退出，代码为 0。
    要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
    按任意键关闭此窗口。 . . .
}

```

改进分析：由于输入数据的大小任意性，如何避免越界问题成为该题的关键，程序采用每乘一次就取其  
后三位，这样就不会产生越界问题。

### 3.输出小数的三位尾数

对任意输入的小于1的并且只有3位尾数的实数，分别求出其各位数字并输出。要求输出的各数字之间要  
空2格。例如，输入为0.368时，输出结果为‘0 3 6 8’。

```

void function()
{
    double a;
    int b[4]{};
    cin>>a;
    a*=10000;
    for(int i = 1 ; i < 4 ; i++)
    {
        b[i] = int(a/pow(10,4-i))%10;
    }
    for(int i = 0 ; i < 4 ; i++)
    {
        cout<<b[i]<<"  ";
    }
}

int main()
{
    function();
    return 0;
}

```

输入：

0.459

输出：

```
7
8
9 double a;
10 int b[4];
11 cin >> a;
12
13
14
15
16
17
18
19
20
21
22
```

算法设计技巧与思路分析：

由于是计算机在储存浮点数的时候并不是输进去多少就是多少的，比如你输入0.368，答案确是 0，3，6，7，原因是在某一步小数之间的减法运算中，储存的小数变为0.36799...并不是0.368，故我们此题乘以1000对整数进行操作

## 4.用递归x形式编程求函数 $H_n(x)$

$H_n(x)$ 定义如下：

$$\begin{cases} H_0(x) = 1 & \leftarrow \\ H_1(x) = 2x & \leftarrow \\ H_n(x) = 2x H_{n-1}(x) - 2(n-1) H_{n-2}(x) & n > 1 \leftarrow \end{cases}$$

递归形式：

```
#include <iostream>
using namespace std;
int function(int n,int x)
{
    if (n == 0)
    {
        return 1;
    }
    else if (n==1)
    {
        return 2 * x;
    }
    else
    {
        return 2 * x * function(n - 1, x) - 2 * (n - 1) * function(n - 2, x);
    }
}
int main()
{
    int n, x;
    cin >> n >> x;
    cout<<function(n,x);
    return 0;
}
```

```
}
```

输入1:

```
0 1231741241
```

输出1:

```
1
```

输入2:

```
1 500
```

输出2:

```
1000
```

输入3:

```
5 10
```

输出3:

```
3041120
```

总结：本题采用递归思想解决复杂问题，对我们将来将复杂问题简单化的思想提供了很好的帮助

---

## 4.实验总结：

这次实验的题型很丰富、全面，更好地巩固了我对c++乃至编程思想的熟悉的程度，在实验中，我更加体会到编程语言的严谨，我经常因为大小写，还有一些拼写错误，导致程序不能运行，而这种错误往往自己需要检查很长很长时间，因为实在是太细微。同时我还更加熟悉的一些算法的优劣性，条件运算符，选择结构，本身它们并没有优劣区分，只是在不同的应用场景下，它们有着不同的优越性。同时我在思考程序算法的时候，经常在纸上进行演算，步骤分析，这样有利于我更好地分析程序，写出更高质量地代码，这个习惯是我应该以后继续保持并好好利用的。

在专业知识方面，我学到了尽量少用条件运算符，静态变量和变量位置有时候能对一个程序造成很大很大的影响，同时cmath库中还有许多实用的数学函数等着我去探索和学习，有规律的数字和图形，我们可以通过循环，以及循环的嵌套来实现。

编程语言的学习是计算机之路上最容易，最基础，但却又是最重要的一步，只有打好基础，我们才能更好地走稳以后的计算机之路。

---

## 5.参考资料

[\(40条消息\) markdown中插入数学公式（全） LeonSUST的博客-CSDN博客md 公式](#)

[\(40条消息\) 四种求最大公约数算法hi你好-CSDN博客 求最大公约数的方法有哪些](#)

[\(41条消息\) cmath用法大全 Quns的博客-CSDN博客](#)

[\(41条消息\) C++函数的定义与使用跟着BOSS有肉吃-CSDN博客c++子函数](#)