实践9小结

- 循环结构实现的算法，理论上都可以采用递归函数实现。设计一个递归函数：
  - 1. 确定功能；2. 终止条件；3. 函数体算法实现。
- 递归法不一定比迭代法效率更高，对于计算机程序而言，递归通常需要更多的运行时间和内存开销。
- 但是递归可以使某些难以采用简单循环解决的复杂问题得到更自然，直接，简单的解决方案，如汉诺塔问题，八皇后问题。

实践9小结　　*正整数n的正逆序输出*

```
void backward (int n, bool isBW)
{
    if(n>=10)
    {
        if(isBW)  cout<<n%10; //isBW为true时，逆序输出
        backward(n/10, isBW);
        if( !isBW) cout<<n%10; //isBW为false时，正序输出
    }
    else cout<<n%10;
}
```

实践9小结

反复输入直至获得正整数

```
//用while循环实现
int setNumWhile(){
    int a;
    cout << "请输入一个正整数：" << endl;
    cin >> a;
    while(a <= 0) {
        cout << "您输入的数值不符合规范（要求：正整数)" << endl;
        cin>>a;
    }
    return a;
}
```

```
//用递归函数实现
int setNumRecursive(){
    int a;
    cout << "请输入一个正整数：" << endl;
    cin >> a;
    if (a <= 0) {
        cout << "您输入的数值不符合规范（要求：正整数)" << endl;
        setNumRecursive();
    }
    else return a;
}
```

实践9小结 *实验八，3.改进求组合数算法，递归实现——考虑好该如何终止递归，然后按公式实现算法。*

$C_n^m = n!/(m!(n-m)!)$ =(n*(n-1)*…*(n-m+1))/m!

=(n/1)*(n-1)/2* …*(n-(m-2))/(m-1)* (n-(m-1))/m= C^{m-1}_n*(n-(m-1))/m

```
int comImprove(int m, int n)
{
        int com;
        if(0==m) com=1;
        else if(1==m) com=n;
        else com=comImprove(m-1, n) * (n-(m-1)) / m;
        return com;
}
```

终止条件 $C_n^0 = 1$ $C_n^1 = n$

```
else com= (n-(m-1)) / m * comImprove(m-1, n);
```

实践9小结

实验八，2.欧几里德法求最大公约数。

```
//辗转相除求余求最大公约数（迭代法）
while (b != 0) {
        temp = a % b;
        a = b;
        b = temp;
}
return a;

//函数递归调用实现
int RecursiveGCD(int a, int b) {
        if (a % b == 0) return b;
        else  return RecursiveGCD(b, a % b);
}
```

实践9小结

- 递归函数的循环控制：
  - 保证可以收敛到递归终止的值。
  - 不是依靠外部循环变量值的改变来控制递归的终止。
  - 递归函数内有输出语句时！ ***P88*,例3.12.**

```
//递归法：按定义，模2后再调用转换函数，直至2^0位
void RecursiveDecToBin(int numDec) {
    if(numDec<2) cout<< numDec%2; //递归终止
    else {
        RecursiveDecToBin(numDec/2);
        cout<< numDec%2;    //递归函数内有输出语句时的分析
    }
}
```

实践9 小结

- 有部分同学理解递归函数调用自身有难度，在实现过程中错误使用函数作为左值：
  - 错误写法Error：fac(n)=n*fac(n-1);
  - 数学函数表达式转换为程序语言表达。
  - 函数可以作为左值，但需要学习引用后学习。
  - P15,左值必须是在内存中可以访问并且可合法修改值的存储单元。（右值需要可以取到确定的值）
  - 传值返回时，函数不可以作为左值，返回前申请的无名临时变量在返回后生命期终止。

Tips: 函数作为左值，上学期仅需了解并理解内存机制。随着下一学期的深入学习和练习，会更好地理解掌握。

实践10 数组初步

- 实验内容：
  - 函数的引用调用(call by reference)
  - 一维、二维数组的定义与初始化
  - 引用、数组作为函数参数

本次作业有难度，但属于必须掌握内容。
若实现有困难，可选择自己能理解的2题，自行完成，
但引用和数组作为函数参数至少各完成一题。

实践10：

- 函数引用调用方式的实践（共3题，至少做1题）。
- 熟练应用一维数组。
- 数组作为函数参数的实践。
- 当形参有引用和数组时，熟练掌握函数的声明、定义、调用。
- 实践内容：实践十一，课本习题5.3~5.4。

程序测试运行结果，注释在程序头部，后文不再提醒。

实践10：提交程序清单1

函数引用调用方式的实践：声明，定义，调用。

编写一个函数，将输入的三个数按升序排序。

函数原型为:

```
void sort(int &n1, int &n2, int &n3);
```

要求：1. 在main函数中随机产生3个整数，

如: num1=3, num2=2, num3=1

2. 首先输出排序前的3个整数到console窗口，

3. 调用sort函数将3个整数排序，

排序后， num1=1, num2=2, num3=3

4. 输出排序后的3个整数到console窗口。

实践10：提交程序清单1

函数引用调用方式的实践：声明，定义，调用。

编写同时计算两数最大公约数和最小公倍数的函数。

函数原型为：

```
void findGcdLcm(int n1, int n2, int &gcd, int &lcm);
```

要求：1. 在main函数中随机产生2个正整数，

2. 首先输出2个整数到console窗口，

3. 调用findGcdLcm函数，

4. 在main函数中调用函数后，将最大公约数和最小公倍数输出到console窗口。

实践10：提交程序清单1

函数引用调用方式的实践：声明，定义，调用。

编写求解一元二次方程实根的函数。函数原型为：

```
void solveQuadratic(double a, double b, double c,
                    double &delta, double &r1, double &r2);
```

要求：1. 在main函数中随机产生3个小数点后有2位的小数作为方程系数abc(随机产生整数后除100，为减少手动验算的工作量，可约束整数部分也是至多2位)，

2. 首先输出3个系数到console窗口，

3. 调用函数后，根据判别式显示计算结果。如果delta>0，显示2个根；如果delta=0，显示1个根；如果delta<0，显示“方程没有实根”。

注意：实型数判零不可以使用 if(delta==0)。

提高：也可以编写程序使用割线法求根法验算求根结果。

实践10：

自选完成，课本例4.5和实验十一范例2，
体会函数作为左值(即返回引用)的使用。

本题作为提高题，自选完成，不提交程序。
函数作为左值，上学期不作考核要求。随着下一学期的深入学习和练习，会更好地理解，熟练地掌握。

实践10：提交程序清单2

习题3.5，数组作为函数参数的练习，设计函数，返回正整数num从右边开始的第k位数字的值。

要求：可支持的整数num位数为1~20位(即超出int数据范围)。

要求：函数原型为：int digit(char num[], int k);

要求：若k值不合理，要求函数可以识别，并返回-1。

注意：此函数定义不够稳健，需要保证在调用时输入的字符串必须有终止符。

思考一下，如果从键盘输入num，如何设计输入代码使得用户体验好一些。

实践10：提交程序清单3

习题5.3，用二维数组作为函数参数打印杨辉三角。

函数原型为：

```
const int LINE=10;
int YHtri[LINE][LINE]={0};
void creatYHtri2D(int [][LINE], int =LINE);
void printYHtri2D(int [][LINE], int =LINE);
```

要求：打印杨辉三角10行。

提示：不限算法，根据杨辉三角的各种属性产生都可以。

输出：等腰或直角三角形都可以，输出范例见后2页。

期待看到多样化的算法!!!

Yang Hui's triangle　1261年《详解九章算法》

Pascal's triangle　1654年

百度百科的算法结果

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

直角三角形

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

实践10：提交程序清单3

习题5.4，用一维数组作为函数参数的练习，用一维数组实现矩阵转置与矩阵相乘。

思考：为什么一维数组可以实现更通用的矩阵算法。

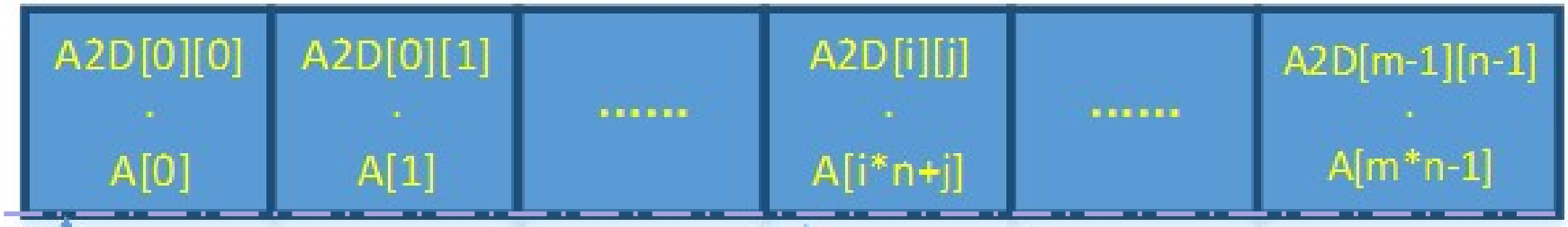
函数原型：

```
void createMatrix(int data[], int length, int min, int max);
void outputMatrix(const int data[], int row, int column);
void MatrixTrans(const int A[], int AT[], int row, int column);
void MatrixMulti(int result[], const int AT[], const int B[], int row, int column1, int column2);
```

提示：下划线参数均为将数据返回主调函数的数组。

二维数组A2D等效一维数组A访问方式

数组下标访问法



A2D [m][n]

.

A [m*n]

*(A2D+i)+j

或 &A2D[0][0]+(i*n+j)

.

A+(i*n+j)

或 &A[0]+(i*n+j)

指针访问法

*createMatrix*函数产生范围在[min,max]之间的随机整数作为矩阵元素，测试时可设定min=max=1，逐步增加数据复杂度验证。

测试范例1例

```
const int ROW     = 3;
const int COLUMN1 = 6;
const int COLUMN2 = 7;
int matrix1[ROW*COLUMN1]={0};
int matrix2[ROW*COLUMN2]={0};
createMatrix(matrix1, ROW*COLUMN1, 1, 1);
createMatrix(matrix2, ROW*COLUMN2, 1, 1);
int middle[COLUMN1*ROW]={0};
int result[COLUMN1*COLUMN2]={0};
MatrixTrans(matrix1, middle, ROW, COLUMN1);
MatrixMulti(result, middle, matrix2, ROW, COLUMN1, COLUMN2);
```

最简范例数据测试结果

Matrix A:

1 1 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1

Matrix B:

1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1

After Transpose:

1 1 1
1 1 1
1 1 1
1 1 1
1 1 1
1 1 1

A^T × B:

3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3
3 3 3 3 3 3 3

代码完成后，也可以用例5.5数据测试程序是否正确，此时不调用随机产生矩阵函数，直接将矩阵按例5.5初始化：

```
const int ROW = 3;
const int COLUMN1 = 6;
const int COLUMN2 = 4;
int matrix1[ROW*COLUMN1]={8,10,12,23,1,3,
    5,7,9,2,4,6,  34,45,56,2,4,6};
int matrix2[ROW*COLUMN2]={3,2,1,0,
     -1,-2,9,8,   7,6,5,4};
```

例5.5数据测试结果

Matrix A:

8 10 12 23 1 3
5 7 9 2 4 6
34 45 56 2 4 6

After Transpose:

8 5 34
10 7 45
12 9 56
23 2 2
1 4 4
3 6 6

Matrix B:

3 2 1 0
-1 -2 9 8
7 6 5 4

A^T × B:

257 210 223 176
338 276 298 236
419 342 373 296
81 54 51 24
27 18 57 48
45 30 87 72

重要！！！

提交的文件名格式：

Exp10_学号_实验名.cpp

注意:1）下划线

2）学号别漏掉

3）各种名字(变量名，文件名)的可读性!

4）每个实验只要提交一个程序源文件，即.cpp文件。

附加题也用*Exp10_学号_实验名.cpp*格式提交！