# LECTURE 8 递归、文件读写

工学院18-19学年秋季学期计算概论（邓习峰班）课后辅导

讲师：陈婉雯

日期：2018/12/15

# 什么是递归?

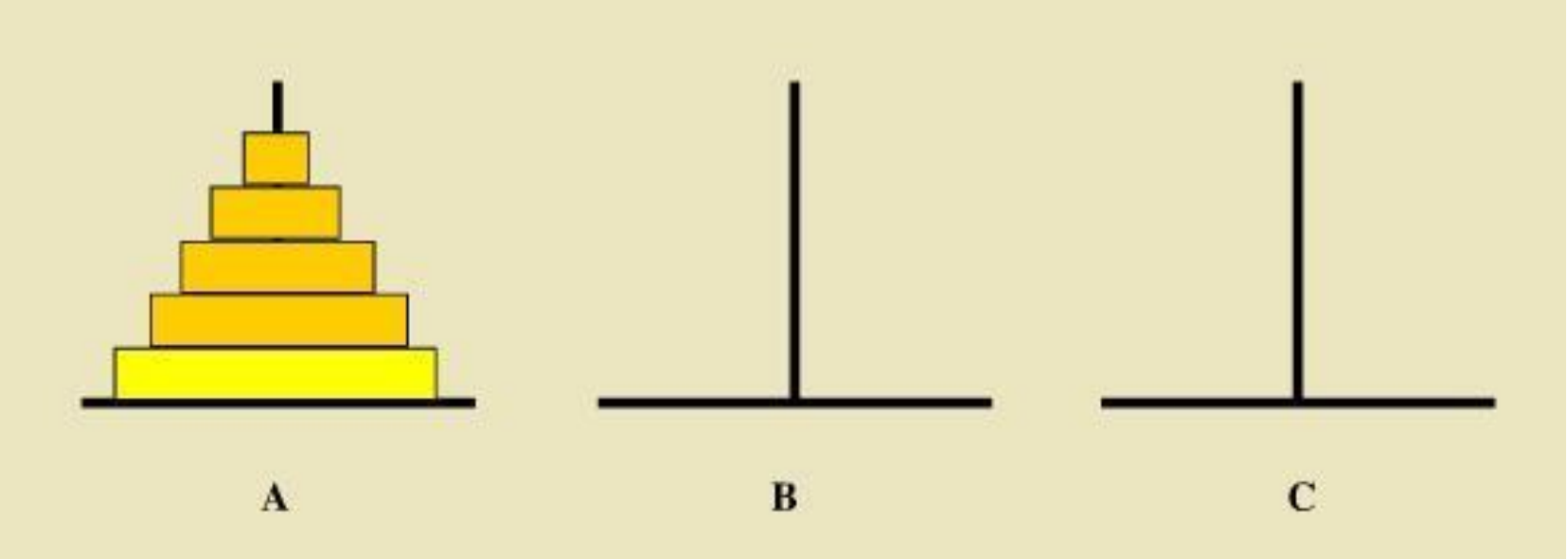
- 递归的定义：参见“递归”。
- 从前有座山，山里有座庙，庙里有个老和尚在讲故事，老和尚在讲：从前有座山，山里有座庙.....

- 递归是一种思考问题、描述问题的方法。一个基本的思想就是，把一个复杂问题化为一系列简单问题的重复。
- 递归还有一个特点，就是问题的规模和解决问题的方法没有关系，或者只是一个参数没有很大的影响。递归所做的，就是把复杂的问题一级一级的展开，使得每一级的处理方法都一模一样。
- 数学归纳法

- 每次递归都能将问题规模缩减下来。
- 递归终止条件，递归必须有个出口，不然会陷入无限递归当中。
- 顺着思路，将递归问题细分为更小的递归问题，然后再进行递归调用。

# Hanoi塔问题

- 移动次数：2^n - 1



# 经典问题

- 用递归求解n!
- 用递归求Fibonacci数列
- 编写一个函数，可以分别打印一个整数十进制的每一位
- 编写一个函数实现n^k
- 不允许创建临时变量求字符串长度，实现strlen的模拟
- 输出全排列
- 逆序打印字符串

# 文件

- 文件是数据源的一种，最主要的作用是保存数据。
- 操作文件的正确流程为：打开文件 --> 读写文件 --> 关闭文件。
  - 文件在进行读写操作之前要先打开，使用完毕要关闭。
  - 打开文件，就是获取文件的有关信息，例如文件名、文件状态、当前读写位置等，这些信息会被保存到一个 FILE 类型的结构体变量中
  - 关闭文件就是断开与文件之间的联系，释放结构体变量，同时禁止再对该文件进行操作
- 数据在文件和内存之间传递的过程叫做文件流，类似水从一个地方流动到另一个地方。数据从文件复制到内存的过程叫做输入流，从内存保存到文件的过程叫做输出流。

# 二进制文件和ASCII文件

- 文本（ASCII）文件：文件中存储的都是字符（ASCII的形式）
- 二进制文件：存储原始的二进制数据
- 例子：
  - 比如存储“123”
  - 在文本文件中存储的为：31 32 33 （每个字符的ASCII的形式），无论“123”是整型数据还是字符串
  - 在二进制文件中的存储方式和“123”的类型有关：
  - 一个字符串123，存储形式是将其ASCII码转换为二进制的形式：其ASCII是31 32 33然后将该ASCII码转换为二进制，也就是：00011111 00100000 00100001 的形式。
  - 一个整型数据123，存储的是123的二进制形式：00000000 00000000 00000000 01111011。

# 文件打开、关闭

- 打开文件： fopen( )
  - 创建一个新的文件或者打开一个已有的文件
  - 会初始化类型 FILE 的一个对象，类型 FILE 包含了所有用来控制流的必要的信息。
  - FILE *fopen( const char * filename, const char * mode );
  - filename 是字符串，用来命名文件
  - mode为访问模式

- 关闭文件：
  - fclose( FILE *fp );

| 模式 | 描述 |
| --- | --- |
| r | 打开一个已有的文本文件，允许读取文件。 |
| w | 打开一个文本文件，允许写入文件。如果文件不存在，则会创建一个新文件。在这里，您的程序会从文件的开头写入内容。如果文件存在，则该会被截断为零长度，重新写入。 |
| a | 打开一个文本文件，以追加模式写入文件。如果文件不存在，则会创建一个新文件。在这里，您的程序会在已有的文件内容中追加内容。 |
| r+ | 打开一个文本文件，允许读写文件。 |
| w+ | 打开一个文本文件，允许读写文件。如果文件已存在，则文件会被截断为零长度，如果文件不存在，则会创建一个新文件。 |
| a+ | 打开一个文本文件，允许读写文件。如果文件不存在，则会创建一个新文件。读取会从文件的开头开始，写入则只能是追加模式。 |

如果处理的是二进制文件，则需使用下面的访问模式来取代上面的访问模式：

```
"rb", "wb", "ab", "rb+", "r+b", "wb+", "w+b", "ab+", "a+b"
```

```
第1行 #include<stdio.h>
第2行 #include<stdlib.h>
第3行
第4行 int main(){
第5行     /*pWriteFile为FILE型指针，指向操作的文件*/
第6行     FILE *pWriteFile=fopen("D:\\myData\\createNewData.txt","w");
第7行     if (pWriteFile==NULL){
第8行         printf("文件打开错误，请检查文件位置和名称是否正确。");
第9行         exit(1);
第10行    }
第11行    int Sum=0;
第12行    for(int i=1;i<=100;++i){
第13行        Sum+=i;
第14行        fprintf(pWriteFile,"%d,%d\n",i,Sum);
第15行    }
第16行    fclose(pWriteFile);
第17行
第18行    return 0;
第19行}
```

避免对打开失败的文件进行操作导致程序异常中止

用完文件之后关闭是一个好习惯

# 文件读写

- 多种文件读写的函数：
  - 字符读写函数 ：fgetc和fputc
  - 字符串读写函数：fgets和fputs
  - 数据块读写函数：fread和fwrite
  - 格式化读写函数：fscanf和fprinf

- 使用以上函数都要求包含头文件stdio.h。

# 文件位置指针

- 文件指针 (FILE *) 是指向整个文件的，须在程序中定义说明，只要不重新赋值，文件指针的值是不变的。
- 文件内部的位置指针用以指示文件内部的当前读写位置，每读写一次，该指针均向后移动，它不需在程序中定义说明，而是由系统自动设置的。

# 字符读写函数

- 以字符（字节）为单位的读写函数。 每次可从文件读出或向文件写入一个字符。
- 从指定的文件中读一个字符：
  - 字符变量=fgetc(文件指针);
  - ch=fgetc(fp); //从打开的文件fp中读取一个字符并送入ch中
- 把一个字符写入指定的文件中：
  - fputc( 字符量, 文件指针 );
  - 待写入的字符量可以是字符常量或变量
  - fputc('a',fp); //把字符a写入fp所指向的文件中

- 在fgetc函数调用中，读取的文件必须是以读或读写方式打开的。
- 读取字符的结果也可以不向字符变量赋值。
- 例如：fgetc(fp); 但是读出的字符不能保存。
- 每写入一个字符，文件内部位置指针向后移动一个字节。
- fputc函数有一个返回值，如写入成功则返回写入的字符，否则返回一个EOF。可用此来判断写入是否成功。

```
第1行  #include<stdio.h>
第2行  #include<stdlib.h>
第3行
第4行  int main(){
第5行      FILE *prFile=fopen("myNewData.txt","r");
第6行
       if (prFile==NULL){
第7行      printf("文件打开错误，请检查文件位置和名称是否正确。");
第8行      exit(1);
第9行  }
第10行 char ch;
第11行 while((ch=fgetc(prFile))!=EOF){
第12行     printf("%c",ch);
第13行 }
第14行 fclose(prFile);
第15行 return 0;
第16行 }
```

运算优先级：
ch=fgetc(prFile);
再判断ch是否为EOF（即文件是否结束）

# 字符串读写函数

- 从指定的文件中读一个字符串到字符数组中
  - fgets(字符数组名,n,文件指针);
  - 其中的n是一个正整数。表示从文件中读出的字符串不超过 n-1个字符。在读入的最后一个字符后加上串结束标志'\0'。
  - fgets(str,n,fp); //从fp所指的文件中读出n-1个字符送入字符数组str中
- 向指定的文件写入一个字符串
  - fputs(字符串,文件指针);
  - 其中字符串可以是字符串常量，也可以是字符数组名或指针变量
  - fputs(“abcd“,fp); //把字符串 “abcd”写入fp所指的文件之中

- fgets函数在读出n-1个字符之前，如遇到了换行符或EOF，则读出结束。
- fgets函数返回值是字符数组的首地址。

```
第1行  #include<stdio.h>
第2行  #define MAXSIZE 512
第3行
第4行  int main(){
第5行      FILE *prFile=fopen("AutumnWind.txt","r");/*AutumnWind.txt李白的秋风词*/
第6行      if (prFile==NULL){
第7行          printf("文件打开错误，请检查文件位置和名称是否正确。");
第8行          return -1;
第9行      }
第10行     char strLine[MAXSIZE];
第11行     int i=0;
第12行     while(!feof(prFile)){
第13行         fgets(strLine,MAXSIZE,prFile);
第14行         printf("%s",strLine);
第15行         i++;
第16行     }
第17行     fclose(prFile);
第18行     return 0;
第19行 }
```

# 格式化读写函数

- fscanf函数，fprintf函数与前面使用的scanf和printf 函数的功能相似，都是格式化读写函数。两者的区别在于fscanf函数和fprintf函数的读写对象不是键盘和显示器，而是磁盘文件。
- 调用格式：
  - fscanf(文件指针,格式字符串,输入表列);
  - fprintf(文件指针,格式字符串,输出表列);
  - fscanf(fp,"%d%s",&i,s);
  - fprintf(fp,"%d%c",j,ch);
- 注意事项和scanf、printf差不多

# 数据块读写函数

- 读数据块函数：
  - fread(buffer,size,count,fp);
- 写数据块函数：
  - fwrite(buffer,size,count,fp);
  - buffer：指针，在fread函数中，它表示存放输入数据的首地址；在fwrite函数中，它表示存放输出数据的首地址。
  - size：表示数据块的字节数。
  - count：表示要读写的数据块块数。
  - fp：表示文件指针。

  - fread(fa,4,5,fp); //从fp所指的文件中，每次读4个字节(一个实数)送入实数组fa中，连续读5次，即读5个实数到fa中。

- fread, fwrite也可以用来读取文本文件。

```
#include<stdio.h>
struct UpDown{
    int Up,Down;
};
int main(){
    FILE *pwFile=fopen("UpDown.Dat","wb");
    if (pwFile==NULL){
        printf("文件打开错误，请检查文件位置和名称是否正确。");
        return -1;
    }
    struct UpDown myUD;
    for(int i=1;i<10;++i){
        myUD.Up=i;
        myUD.Down=i+1;
        fwrite(&myUD,sizeof(myUD),1,pwFile);
    }
    fclose(pwFile);
    /*将写入数据读出*/
    FILE *prFile=fopen("UpDown.Dat","rb");
    while(!feof(prFile)){
        fread(&myUD,sizeof(myUD),1,prFile);
        printf("%d/%d ",myUD.Up,myUD.Down);
    }
    fclose(prFile);
    return 0;
}
```

```c
#include <stdio.h>
#include <string.h>

int main(){
    char c[20];
    int MAX=20;
    FILE *fp, *ofp;
    fp=fopen("test.txt","r");
    ofp=fopen("output.txt","w");
    fread(c,sizeof(char),20,fp);
    printf("%s",c);
    fwrite(c,sizeof(char),20,ofp);
    fprintf(ofp,"\n");
    fwrite(c,sizeof(char),strlen(c)-1,ofp);
    fclose(fp);
    fclose(ofp);
    return 0;
}
```

注意两个fwrite里面输出数据块长度不同
输出结果有什么不同？

test.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

Hello world!

output.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

Hello world!
7
Hello world!

# 文件定位

- fseek()用来移动文件流的读写位置
  - int fseek(FILE * stream, long offset, int whence);
  - 参数stream 为已打开的文件指针；参数offset 为根据参数whence来移动读写位置的位移数。
  - 参数 whence 为下列其中一种：
  - SEEK_SET 从距文件开头offset 位移量为新的读写位置
  - SEEK_CUR 以目前的读写位置往后增加offset 个位移量
  - SEEK_END 将读写位置指向文件尾后再增加offset 个位移量
  - 当whence值为SEEK_CUR或SEEK_END 时, 参数offset允许为负

  - 将读写位置移动到文件开头 fseek(FILE *stream, 0, SEEK_SET);
  - 将读写位置移动到文件尾 fseek(FILE *stream, 0, 0SEEK_END);

# 文件定位

- ftell()用来获取文件读写指针的当前位置
  - long ftell(FILE * stream);
  - stream 为已打开的文件指针
  - 成功则返回当前的读写位置，失败返回 -1

```c
#include <stdio.h>

int main ()
{
   FILE *fp;
   int len;

   fp = fopen("file.txt", "r");
   if( fp == NULL )
   {
      perror ("打开文件错误");
      return(-1);
   }
   fseek(fp, 0, SEEK_END);

   len = ftell(fp);
   fclose(fp);

   printf("file.txt 的总大小 = %d 字节\n", len);

   return(0);
}
```

# 文件定位

- rewind()将文件指针重新指向文件的开头
  - void rewind(FILE * stream);
  - stream为以打开文件的指针
  - 相当于fseek(stream, 0, SEEK_SET)

```
第13行     int Offset;/*定义偏移量值*/
第14行     printf("请输入读取数据编号(从0开始)： ");
第15行     scanf("%d",&Offset);
第16行     fseek(prFile,Offset*sizeof(struct UpDown),SEEK_SET);/*按指定位置定位*/
第17行     fread(&myUD,sizeof(struct UpDown),1,prFile);/*从指定位置读取数据*/
第18行     printf("%d/%d\n",myUD.Up,myUD.Down);
第19行     printf("文件指针当前位置=%ld\n",ftell(prFile));/*获得文件指针的当前位置*/
第20行
第21行     rewind(prFile);/*文件指针再次指向开始*/
第22行     fread(&myUD,sizeof(struct UpDown),1,prFile);/*从指定位置读取数据*/
第23行     printf("%d/%d\n",myUD.Up,myUD.Down);/*输出第一个数*/
第24行     printf("文件指针当前位置=%ld\n",ftell(prFile));
第25行
第26行     fclose(prFile);
第27行     return 0;
第28行   }
```