数据结构实验

肖清

课程教学目的

在已学习“数据结构”理论的基础上，通过实验帮助同学们巩固数据结构基础知识、培养数据抽象能力、数据结构设计能力、程序设计能力，进而提高实践能力。

课程要求

掌握线性表、栈、队列、二叉树、图的逻辑结构、存储结构及其操作算法的实现和应用；了解查找、排序的主要算法的实现和应用。

期末采用大作业形式进行课程考核；

平时成绩涵盖课堂出勤、课堂表现、实验成绩(包括实验报告和程序)。

课程QQ群： 834743189(1班)、 836341667(2班)

邮箱： 2912385135@qq.com(1班)、 qilong221b@163.com(2班)

实验一　函数

一、实验目的

1、复习变量、数据类型、语句、函数；

2、掌握函数的参数和值；

3、了解递归。

引例1

```
int main()
{    int n_a=5,n_b=10;//A组、B组学生的人数

    int score_a[5]={56,80,76,63,90};//A组学生的分数数组
    int score_b[10]={96,88,36,58,66,78,89,100,93,73};//B组学生的分数数组

    char grade_a[5]={' ',' ',' ',' ',' '};//A组学生的等级数组（函数参数）
    char grade_b[10]={' ',' ',' ',' ',' ',' ',' ',' ',' ',' '};//B组学生的等级数组（函数参数）

    float average_p_a=0,average_p_b=0;//A组、B组学生的平均分（函数参数）

    float average_v_a=Score_Grade_Average(n_a,score_a,grade_a,average_p_a);//A组学生的平均分（函数值）
    float average_v_b=Score_Grade_Average(n_b,score_b,grade_b,average_p_b);//B组学生的平均分（函数值）

    Print(n_a,score_a,grade_a,average_p_a,average_v_a);
    printf("\n\n\n");
    Print(n_b,score_b,grade_b,average_p_b,average_v_b);
}
```

引例1

```
float Score_Grade_Average(int n,int score[ ],char grade[ ],float average)
//输入n个学生的0~100之间的分数数组score，转换分数为等级并计算平均分，输出相应的等级数组（函数参数grade）及平均分（函数参数average和函数值）。
//关注函数参数grade和average的变化！
{   int i;    float sum;
    for(sum=0,i=0;i<n;i++)
    {   sum=sum+score[i];
        switch(score[i]/10)
        {   case 10:
            case 9: grade[i]='A'; break;
            case 8:
            case 7: grade[i]='B'; break;
            case 6: grade[i]='C'; break;
            default:
grade[i]='D';   }    }
    average=sum/n;
    return average;}
```

```
void Print(int n,int score[],char grade[],float average_p,float average_v)
//输入并输出n个学生的分数数组score、等级数组grade、平均分average_p和average_v。
{   int i;
    printf("这组学生的人数是：%d\n\n平均分是：%.2f（函数参数）\t还是：%.2f（函数值）\n\n各个学生的分数与等级（函数参数）如下：
\n",n,average_p,average_v);
    for(i=0;i<n;i++)
        printf("第%d个学生：\t分数为%d\t等级为%c\n",i+1,score[i],grade[i]);
}
```

引例1

思考1：
函数的好处是什么？
在什么情况下，将处理步骤写成函数较好？

思考2：
如何从函数中获得多个处理结果？

引例2

```
int main()
{   printf("采用循环计算5!=%ld\n",Factorial_Loop(5));
    printf("采用循环计算30!=%ld\n",Factorial_Loop(30));
    printf("\n");
    printf("采用递归计算5!=%ld\n",Factorial_Recursion(5));
    printf("采用递归计算30!=%ld\n",Factorial_Recursion(30));
}
```

引例2

```
long Factorial_Loop(int n)
//采用循环计算n的阶乘n!
//输入n，计算并输出n!
{   int i;
    long result;

for(result=1,i=2;i<=n;i++)
        result=result*i;
    return result;
}
```

```
long Factorial_Recursion(int n)
//采用递归计算n的阶乘n!
//输入n，计算并输出n!
{   if (n==1)
        return 1;
    else
        return n*Factorial_Recursion(n-1);
}
```

引例2

思考1：
递归方法与非递归方法的好处是什么？
在什么情况下，采用递归方法较好？
而在什么情况下，采用非递归方法较好？

思考2：
在现实生活中有哪些问题可采用递归方法？

实验一 函数

二、实验内容

1、(必做题)采用函数统计学生成绩：输入学生的成绩，计算并输出这些学生的最低分、最高分、平均分。

2、(必做题)采用递归和非递归方法计算k阶裴波那契序列的第n项的值，序列定义如下：

$f_0=0,\ f_1=0,\ \cdots,\ f_{k-2}=0,\ f_{k-1}=1,$

$f_n=f_{n-1}+f_{n-2}+\cdots+f_{n-k}\quad (n>=k)$

要求：输入k(1<=k<=5)和n(0<=n<=30)，输出f_n。

实验一 函数

二、实验内容

3、（二选一）采用递归和非递归方法求解汉诺塔问题，问题描述如下：

有三根柱子A、B、C，在柱子A上从下向上有n个从大到小的圆盘，在柱子B和C上没有圆盘，现需将柱子A上的所有圆盘移到柱子C上，可以借助柱子B，要求每次只能移动一个圆盘，每根柱子上的圆盘只能大的在下，小的在上。

要求：输入n，输出移动步骤。

实验一 函数

二、实验内容

4、（选做题）采用递归和非递归方法求解，问题描述如下：

有一群猴子摘了些桃子放在仓库，每天吃掉所有桃子的一半还要多吃一颗。第n天时，还剩下m颗桃子，问一开始放入仓库的桃子几颗？

要求：可循环输入n，m求解，直到输入-1时退出程序;输出一开始的桃子总数sum。