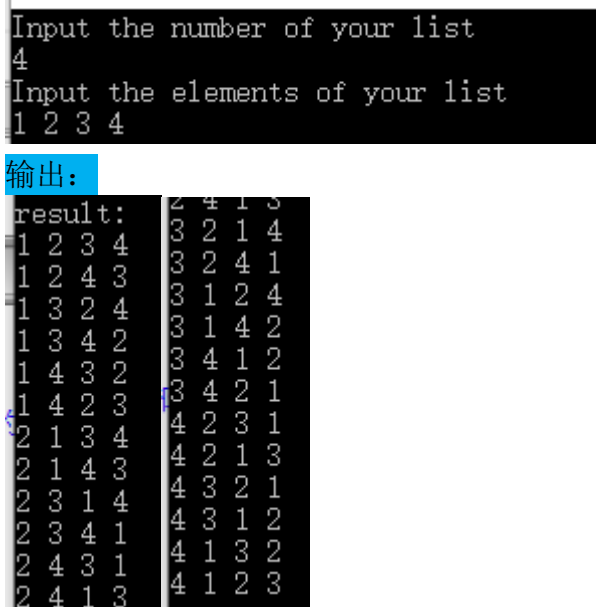


数据结构与算法 课程实验报告

学号：201700130033	姓名：武学伟	班级：2017 级 2 班
实验题目：递归练习		
实验学时：4	实验日期：2018. 10. 6	
实验目的： 1、熟悉开发工具的使用。 2、掌握递归的实现思想。		
软件环境： Win10home, codeblocks		
1. 实验内容（题目内容，输入要求，输出要求） 1) 键盘输入n和n个互不相同的整数，输出n个整数的全排列。 2) 键盘输入n和n个互不相同的整数，输出n个整数的所有子集。		
2. 数据结构与算法描述（整体思路描述，所需要的数据结构与算法） 1) 对于数组[0:n-1], 当只有一个元素时, 排列就是本身. n=1, 集合E只有一个元素e, 用perm[E]表示集合E的所有排列, perm(E)=e。 当 n>1 时, perm(E)是一个表, 用 $e_i. perm(X)$ 表示在 perm(X) 中的每个排列加上前缀 e_i 之后的排列表. perm(E)= $e_1. perm(E_1), e_2. perm(E_2), \dots, e_n. perm(E_n)$, 这是 perm(E) 的递归定义 用 list[0:k-1] 表示前缀, 用 list[k:m] 表示后缀, 当 k=m 时, 输出排列, 当 k<m 时, 定义 i 从 k 至 m 遍历, 交换 list[k] 和 list[i] 的值, 调用递归, 再次交换, 恢复初始状态。 2) 对于数组 a[0:n-1], 定义伴随数组 m[0:n-1], 用 0 和 1 来确定目标数组的元素是否出现 定义变量 now, 表示 a 中元素位置, 定义变量 sum 表示 a 中元素总数 初始化 now=0, 令 m[now] 分别为 0 和 1, 如果 now=sum, 输出, 否则令 now+1 输出: 如果 m[i]=1, 输出 a[i], 否则输出" "。 3. 测试结果（测试输入，测试输出，结果分析） 1) 输入: 4 1 2 3 4  输出: <pre> result: 1 2 3 4 2 4 1 3 1 2 4 3 3 2 1 4 1 2 3 4 3 2 4 1 1 3 2 4 3 1 2 4 1 3 4 2 3 1 4 2 1 4 3 2 3 4 1 2 1 4 2 3 3 4 2 1 2 1 3 4 4 2 3 1 2 1 4 3 4 2 1 3 2 3 1 4 4 3 2 1 2 3 4 1 4 3 1 2 2 4 3 1 4 1 3 2 2 4 1 3 4 1 2 3 </pre> 2) 输入: 5		

1 2 3 4 5

```
Input the number of the array
5
Input the member of the array
1 2 3 4 5
```

输出:

```
Result: {2345}
{12345} {234}
{1234} {235}
{1235} {23}
{123} {245}
{1245} {24}
{124} {25}
{125} {2}
{12} {345}
{1345} {34}
{134} {35}
{135} {3}
{13} {45}
{145} {4}
{14} {5}
{15} {0}
{1}
```

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

- 1) 一组数据的全排列是 $n!$ 种，验证一中输入和输出的数据，符合集合元素的全排列的结果。
- 2) 一组数据的全部子集数是 2^n 种，验证二中输入和输出的数据，符合集合元素的全部子集的结果。

5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

```
/*Ex1_1.cpp*/
```

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

```
template<class T>
void Swap(T &a, T &b)
{
    T temp;
    temp = a;
    a = b;
    b = temp;
}
```

```
//要求不使用 STL 函数，编写 Swap 函数用于两数之间的交换，等价于 stl 函数 swap
```

```
template<class T>
void perm(T list[], int k, int m) //生成指定列表的全排列
{
    if (k==m) //当 k 等于 m 时，后缀只有一种排列，此时就可以输出全部的数组 list[0:m]
    {
        for (int i=0; i<=m; i++)
            cout << list[i] << " ";
        cout << endl;
    }
}
```

```

else //当不等时，有多个排列
    for (int i=k; i<=m; i++)
    {
        Swap(list[k],list[i]); //将 list[i]与 list[k]交换
        perm(list, k+1, m);
        Swap(list[k],list[i]); //恢复调用前的状态
    }
}

```

```

int main()
{
    int n;
    cout << "Input the number of your list" << endl;
    cin >> n; //输入列表元素个数
    int a[n];
    cout << "Input the elements of your list" << endl;
    for (int i=0; i<n; i++)
        cin >> a[i]; //输入列表元素
    cout << "result:" << endl;
    perm(a,0,n-1); //输出全排列
    return 0;
}

```

```

/*Ex1_2. cpp*/

```

```

#include <iostream>
using namespace std;

```

```

template <class T>
void Collection_Arrage (T a[], int m[], int now, int sum) //a 是指定的数组，m 是标记
的数组，now 是数组当前位，sum 数组元素个数
{
    if (now == sum) //判断是否已经到达数组末位
    {
        cout << "{" ;
        for (int i=0; i<sum; i++)
        {
            if (m[i] == 1) //当标记数组的元素值为 1 时，输出指定数组的元素值，否
            则输出为空格
                cout << a[i];
            else
                cout << " ";
        }
        cout << "}" << endl;
        return;
    }
    m[now] = 1; //分别令各个元素对应的标记数组元素值为 1 或 0
    Collection_Arrage(a, m, now+1, sum);
}

```

```
    m[now] = 0;
    Collection_Arrage(a, m, now+1, sum);
}
int main ()
{
    int n;
    cout << "Input the number of the array" << endl;
    cin >> n;    //输入 n
    int a[n];
    cout << "Input the member of the array" << endl;
    for (int i=0; i<n; i++)
        cin >> a[i];    //输入 n 个元素
    cout << "Result:" << endl;
    int m[n];
    m[n] = 0;
    Collection_Arrage(a, m, 0, n);    //进行 n 个元素的全部子集
    return 0;
}
```