**实验一：表达式求值**

1. **实验目的**

熟练掌握栈，队列等基本操作及其在实际问题中的应用，对于正确的数学表达式求取其值。

1. **实验要求**

实现栈的push,pop等基本操作，及表达式求值的实现

检测表达式的输入是否是正确的数学表达式

对正确的表达式求值，支持加减乘除，小，中，大括号及幂运算

注：

1）数学表达式需支持小数、加减乘除、小括号、中括号、幂运算。为简化逻辑，不需要考虑负数情况。

2）不正确的表达式可能包括：字母、非运算符号、括号不匹配，运算符的排列不符合表达式形式等多种情况。

3）表达式通过命令行参数读取 。

1. **设计思路**

定义两个栈OPTR和OPND分别用来存放运算符和运算数。从命令行获取参数，若参数为运算数则直接压入栈中。

若为运算符，则需要与OPTR栈顶元素比较：若栈顶元素优先级小，则运算符直接入栈。若两者优先级相等，则说明两者为相匹配的括号，则用pop操作将运算符栈栈顶的括号出栈。若栈顶运算符的优先级大，则将栈顶的运算符pop出栈，同时将运算数栈中最上面的两个运算数出栈（若运算数数量不足，则说明运算符的排列不符合表达式的形式，输出ERROR\_2)。两个运算数通过该运算符运算后的结果压入OPND栈中。

不断获取参数，直到参数获取完毕并且OPTR中无运算符时，结束操作，输出OPND的栈顶元素即为表达式的值。

1. **运行结果**



1. **问题与思考**

在编程的过程，在对小数处理的过程中程序设计有些繁琐。

我们在编程的过程中一定要提高程序的鲁棒性，用黑暗森林的思路来考虑问题，每个函数都要考虑异常输入的情况。

**实验二：基于串的模式匹配**

1. **实验目的**

熟练掌握抽象数据类型串的实现，学会使用串及其模式匹配算法解决具体应用问题，从而体会串的特点。

1. **实验要求**

用KMP算法实现模式匹配

1. **设计思路**

首先判定是否合法输入。

若合法输入开始计算模式串Next数组，随后KMP调用模式串进行匹配。

若匹配成功则返回位置，free变量，否则异常输出并free内存。

1. **运行结果**



1. **问题与思考**

若传参过多可以考虑结构体传递参数，尽量避免全局变量和不明意义的数字，对后期理解造成过大困难。

**实验三：基于霍夫曼树的编码/译码**

1. **实验目的**

掌握二叉树的生成与遍历操作，及霍夫曼编码译码的原理

1. **实验要求**

熟练掌握树的操作

基于该霍夫曼树，实现非递归的先序遍历算法，输出该树所有的节点，节点的权值，节点的度和节点所在的层数

在实现时要求霍夫曼树的左孩子节点权值小于右孩子节点权值

1. **设计思路**

本实验构造两个指针HT和HC。HT指向二叉树的所有节点，其中前n位是n个叶子节点，最后一位是根节点，其中每个节点中存放有节点的数据域，权值，父节点位置，左孩子位置，右孩子位置。HC指向一段存指针的地址，这些指针指向输入的n个不同字符的霍夫曼编码。

霍夫曼树的构造过程就是先将所有的叶子节点存入数组，然后在数组中找出权值最小的两个节点构成一棵新树，新树根节点权值为两者权值之和。将该根节点存入数组中，更新两个子节点的父节点位置等相关信息。如此重复，最后成功构造一棵霍夫曼树，数组最后一个元素即为该霍夫曼树根节点。

编码过程就是通过比较命令行参数的字符和HT中节点的数据域，找到二者相同的节点所在的位置i，调用HC[i]输出该字符的霍夫曼编码，一个个比较直至输出最后一个字符的赫夫曼编码。

译码过程就是一位位地取第二个命令行参数的字符，从HT的根节点开始，如果字符为0则指向根节点的指针移向左子树，字符为1则移向右子树，直至到达一个叶子节点，将叶子节点的数据域输出。输出后再重新从根节点开始向下走输出下一个叶子节点的数据域。如果命令行参数获取完毕而指针未指向叶子节点，则译码失败输出ERROR\_3。

我们用先序遍历输出该树所有节点的信息。即从根节点开始，指针指向哪个节点则输出该节点的信息，并将该节点入栈。如果该节点左子树存在，令指针指向左子树。若节点的左子树不存在，则令栈顶元素出栈，指针指向出栈元素右子树。若右子树依然不存在，则继续让栈顶元素出栈，继续让指针指向出栈元素右子树。依次循环直至所有节点都遍历完毕。

1. **运行结果**



1. **问题与思考**

赫夫曼树的编码译码的难点在于赫夫曼树及其权值数组和字符数组的构建。

关于赫夫曼树中子节点的存储顺序我采用的是按其在命令行参数中出现顺序来排列的。并且在获取命令行参数中的字符时，每一个字符都要判断这个字符是否是第一次出现，因此构建赫夫曼树的时候循环套循环，增加了程序的时间复杂度。

程序开辟的空间结束时需要free，开辟空间后也要需要进行指针的判空。树是一种重要的数据结构，其在编码译码，排序查找等多个方面都有着重要的作用。

**实验四：无向图最短路径搜索**

1. **实验目的**

掌握Dijkstra算法的原理

1. **实验要求**

熟练掌握图的操作

针对校园知名建筑平面图进行构建数据结构和算法，通过命令行参数输入任意两建筑的名字，输出最短路径长度和最短路径。

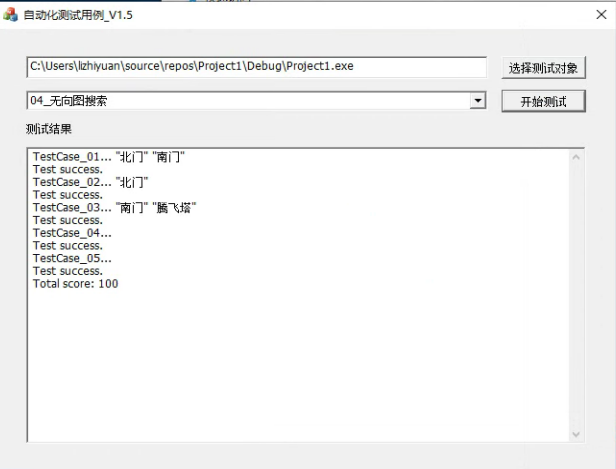
1. **设计思路**

首先根据校园知名建筑平面图构建图G，包括顶点数组和其邻接矩阵。构造数组final,确定顶点是否包含在最短路径中。再构造矩阵P，如果P[v][w]为TRUE，则w为v0到v的最短路径上的顶点。再构造数组D，储存各个顶点到v0最短路径长度。

再运用Dijkstra算法：任意选则一顶点v0，找出与此顶点距离最近的顶点vj,将vj顶点加入final数组，更新v0顶点到其余各顶点的最短距离，即更新D数组的值，更新P矩阵。再找出距离最短的顶点加入顶点数组，不断更新D数组和P矩阵。如此重复，则可以找出所有顶点到v0的最短路径长度，即最终D数组储存的值。

最短路径则可以通过P矩阵与邻接矩阵G.Edge获得，如果顶点i为v0到v1路径上的点，且i与已输出路径的最后一个顶点邻接，则i一定为下一个要输出的顶点。输出i顶点后，将i顶点与上一个顶点的邻接关系断开，并且顶点i成为已输出路径的最后一个顶点。如此重复便可以将最短路径输出。

1. **运行结果**



1. **问题与思考**

最短路径问题的关键便是对Dijkstra算法的理解与运用。合理运用P数组记录路径上的顶点有利于我们将最短路径求出来。

图也是一种非常重要的数据结构。最短路径问题在提供生活便利，降低经济成本等方面的运用也是十分广泛的，比如路线导航，或是管道修建的最小成本等。我们应该要熟练掌握图的两种遍历操作，最小生成树的生成和最短路径的计算等操作，将所学的知识运用到生活中去。

**附：四次实验的cpp源文件**

