实验3: 串和数组

确保把类定义写在string\_array.h中，类实现写在string\_array.cpp中

给出的string\_array.h中的函数声明不要改动，string\_array.h中的内容可以增但不可以删，测试文件将根据已有的函数声明调用，string\_array.cpp中的打印函数不要改动。

将类和函数声明写在书string\_array.h中，类和函数的实现写在string\_array.cpp中，最终提交两个文件：string\_array.h和string\_array.cpp

1. 实现字符串类myStr，存储方式不限。实现以下功能：

* 字符串初始化myStr(const char\*)，提示：字符串以\0结尾
* 字符串销毁~myStr()
* 字符串输出void print()
* 其他必要的成员函数
* 实现myStr的友元函数，串的替换bool replaceStr(myStr& S, const int& start, const myStr& T, const myStr& V)，即要求在主串S中，从位置start开始查找是否存在子串T，若主串S中存在子串T，则用子串V替换子串T，且函数返回1；若主串S中不存在子串T，则函数返回0， start取值从1开始
* kmp辅助数组next的计算，void kmp\_next()
* kmp辅助数组nextVal的计算，void kmp\_nextVal()
* kmp辅助数组next的输出void printNext()
* kmp辅助数组nextVal的输出void printNextVal()
* 实现myStr的友元函数，简单字符串匹配算法int simpleMatch(const myStr& S, const myStr& T)，目标串S和模式串T，求T在S中的位置。匹配失败返回-1，匹配成功返回匹配位置（字符串的位置从1开始）
* 实现myStr的友元函数，改进KMP算法的字符串匹配算法int kmpMatch(const myStr& S, const myStr& T)，目标串S和模式串T，求T在S中的位置。匹配失败返回-1，匹配成功返回匹配位置（字符串的位置从1开始）

输入输出示例：

myStr S("hello TJU!");

myStr T("TJU");

myStr T1("NKU");

myStr T2("future");

int isFind;

isFind=replaceStr(S,8,T1,T2);

cout<<isFind<<endl;//0

isFind=replaceStr(S,1,T, T2);

cout<<isFind<<endl;//1

S.print();//hello future!

isFind=replaceStr(S,1,T1, T2);

cout<<isFind<<endl;//0

myStr matchS("aabaabaabaac");

myStr matchT("aabaac");

isFind=simpleMatch(matchS,matchT);

cout<<isFind<<endl;//7

matchS.kmp\_next();

matchS.kmp\_nextVal();

matchS.printNext();//012123456789

matchS.printNextVal();//002002002009

matchT.kmp\_next();

matchT.kmp\_nextVal();

matchT.printNext();//012123

matchT.printNextVal();//002003

isFind=kmpMatch(matchS,matchT);

cout<<isFind<<endl;//7

1. 实现稀疏矩阵类myMatrix，使用三元组存储稀疏矩阵元素，实现以下功能：

* 初始化稀疏矩阵，myMatrix (const int& rNum, const int& cNum, const int& nNum, const int\*),参数依次为行数、列数、三元组元素个数、三元组初始化数据，数组元素为3的倍数，每3个数一组，分别为（row，col，value）
* 初始化稀疏矩阵myMatrix（）
* 销毁稀疏矩阵，~ myMatrix()
* 其他必要的成员函数
* 实现快速转置算法void FastTransposeSMatrix(myMatrix& T),转置结果存在T中
* 打印矩阵void printMatrix()，打印格式为：行数，列数，元素数

行，列，元素值

……

输入输出示例

int i[]={2,3,10,2,4,30,4,4,25};

myMatrix M(5,6,3,i);

M.printMatrix();//5,6,3

//2,3,10

//2,4,30

//4,4,25

myMatrix T;

M.FastTransposeSMatrix(T);

T.printMatrix();//6,5,3

//3,2,10

//4,2,30

//4,4,25