**数据结构习题3**

**学号： 姓名：**

**1、**4-19

#include <stdio.h>

#include <stdlib.h>

const int MAXSIZE = 1000; // 顺序串最大容量

typedef struct linknode // 定义顺序串

{

char ch[MAXSIZE];

int len;

}SeqString;

void createStr(SeqString \*s, char cstr[]) // 创建顺序串，即将已有的字符串转为顺序串

{

int i;

for(i = 0; cstr[i] != '\0'; i ++ ) // 顺序对应，将字符串中的字符挨个放入顺序串中

s -> ch[i] = cstr[i];

s -> ch[i] = '\0';

s -> len = i; // 记录串长，即为字符串中的元素个数

}

SeqString deleteSubStr(SeqString s, int i, int j) // 删除子串函数

{

SeqString str = {"", 0};

int k;

if(i <= 0 || i > s.len || j < 0 || i + j > s.len + 1) // 检测错误输入，当输入错误时返回空的顺序串

return str;

// 此处将给定的顺序串进行拷贝，但取消对应子串的拷贝，达到删除效果

for(k = 0; k < i - 1; k ++ )

str.ch[k] = s.ch[k];

for(int m = i + j - 1; m < s.len; m ++ )

str.ch[k ++ ] = s.ch[m];

str.len = k; // 新的顺序串元素个数等于拷贝元素个数

return str; // 返回新顺序串

}

void outputStr(SeqString s) // 输出顺序串

{

printf("%s", s.ch);

}

int main()

{

SeqString s;

char str[MAXSIZE];

int i, j;

scanf("%s", str);

createStr(&s, str);

scanf("%d%d",&i, &j);

s = deleteSubStr(s, i, j);

outputStr(s);

return 0;

}

**2、**4-20

#include <stdio.h>

#include <stdlib.h>

const int MAXSIZE = 10000; // 输入字符的最大长度

typedef struct linknode

{

char data;

struct linknode \*next;

}linkstring;

linkstring \*StringAdd(linkstring \*head, char \*string) // 链串添加元素（创建链串）

{

linkstring \*r = head, \*s;

if(r != NULL) // 将当前结点遍历到链表尾，当头结点为空时不需要遍历

while(r -> next) r = r -> next;

for(int i = 0; string[i] != '\0'; i ++ )

{

s = (linkstring \*)malloc(sizeof(linkstring));

s -> data = string[i];

if(r != NULL) // 头结点为空时，将当前结点复制给头结点，否则当前结点加入链表

r -> next = s;

else

head = s;

r = s;

}

r -> next = NULL; // 尾结点指向空

return head; // 返回头结点

}

void CharReplace(linkstring \*head, char bp, char tp) // 字符替换函数

{

for(linkstring\* r = head; r; r = r -> next) // 遍历所有字符，找到相同字符进行替换

if(r -> data == bp)

r -> data = tp;

}

void OutputString(linkstring \*head) // 字符输出函数

{

for(linkstring \*r = head; r; r = r -> next) // 遍历所有字符，进行输出

printf("%c", r -> data);

printf("\n");

}

int main()

{

linkstring \*head = NULL;

char s[MAXSIZE];

char bp, tp;

printf("请输入字符串以创建链串\n");

fgets(s, MAXSIZE, stdin);

head = StringAdd(head, s);

printf("请输入要被替换的字符\n");

scanf("%c", &bp);

getchar();

printf("请输入替换成为的字符\n");

scanf("%c", &tp);

CharReplace(head, bp, tp);

printf("替换后的字符串为：\n");

OutputString(head);

printf("\n");

return 0;

}

**3、**4-26

#include <stdio.h>

#include <stdlib.h>

const int SMAX = 16; // 最大长度

typedef struct

{

int row, col;

int value;

}Node;

typedef struct

{

int rows, cols, nums;

Node data[SMAX];

}TSmatrix; // 三元组储存稀疏矩阵

TSmatrix createMat(int a[][100], int rows, int cols) // 将数组转换为稀疏矩阵

{

TSmatrix t;

t.rows = rows; // 数组的行列数等于稀疏矩阵的行列数

t.cols = cols;

t.nums = 0; // 稀疏矩阵元素初始化为0

for(int i = 0; i < rows; i ++ )

for(int j = 0; j < cols; j ++ )

if(a[i][j] != 0) // 遍历数组，不为0的元素放入稀疏矩阵中

{

t.data[t.nums].row = i;

t.data[t.nums].col = j;

t.data[t.nums].value = a[i][j];

t.nums ++ ;

}

return t;

}

int searchValue(TSmatrix t, int row, int col) // 寻找对应行列数的稀疏矩阵值

{

int k = 0;

while(k <= t.nums && t.data[k].row <= row)

{

if(t.data[k].row == row && t.data[k].col == col) // 找到对应行列处的矩阵值

return t.data[k].value;

k ++ ;

}

return 0; // 若找不到，即稀疏矩阵并未储存该处的值，即为0

}

TSmatrix MatrixMul(TSmatrix A, TSmatrix B) // 矩阵乘法

{

TSmatrix C = {0, 0, 0, 0}; // 初始化C

if(A.cols != B.rows) // 判断是否可以相乘

{

printf("矩阵A列数不等于矩阵B行数,无法相乘");

return C;

}

C.rows = A.rows; // 根据矩阵乘法，得到的矩阵行数等于前矩阵行数，列数等于后矩阵列数

C.cols = B.cols;

int cnt = 0; // 计数，用于给新稀疏矩阵的nums赋值

if(A.nums != 0 && B.nums != 0) // 当两个矩阵不全为零矩阵时，进行乘法

{

for(int row = 0; row <= A.rows; row ++ )

{

for(int col = 0; col <= B.cols; col ++ )

{

// 遍历C的所有行列点，运用矩阵乘法原理，求累加和，不为0时，放入新稀疏矩阵中

int sum = 0;

for(int k = 0; k <= A.rows; k ++ )

sum += searchValue(A, row, k) \* searchValue(B, k, col);

if(sum != 0)

{

C.data[cnt].row = row;

C.data[cnt].col = col;

C.data[cnt].value = sum;

cnt ++ ;

}

}

}

C.nums = cnt; // 元素个数

}

return C; // 当其中有零矩阵时，会返回初始化的C（行列数已更新）

}

void outPut(TSmatrix t) // 输出稀疏矩阵

{

printf("\n\n row col val\n\n");

for(int i = 0; i < t.nums; i ++ )

printf("%5d%6d%6d\n", t.data[i].row + 1, t.data[i].col + 1, t.data[i].value);

}

int main()

{

int a[100][100], b[100][100];

int rowa, cola, rowb, colb;

printf("请输入A B矩阵的行列数与矩阵的值\n");

scanf("%d%d%d%d", &rowa, &cola, &rowb, &colb);

for(int i = 0; i < rowa; i ++ )

for(int j = 0; j < cola; j ++ )

scanf("%d", &a[i][j]);

for(int i = 0; i < rowb; i ++ )

for(int j = 0; j < colb; j ++ )

scanf("%d", &b[i][j]);

TSmatrix A = createMat(a, rowa, cola);

TSmatrix B = createMat(b, rowb, colb);

TSmatrix C = MatrixMul(A, B);

outPut(C);

return 0;

}