4-18

int sameStr(char x[], char y[])

{

int i = 0;

int tag = 1;

if(StrLen(x) != StrLen(y))

return 0;

else

{

while(i < StrLen(x) && tag == 1)

{

if(x[i] != y[i])

tag = 0;

i++;

}

return tag;

}

}

4-19

int StrDelete(char s[], int i, int j)

{

if(i < 0 || i > StrLen(s) - 1)

return 0;

if(j < 0 || j > StrLen(s) - i)

return 0;

for(int k = i + j; k < StrLen(s); k++)

{

s[k - j] = s[k];

}

s[StrLen(s)] = '\0';

return 1;

}

4-20

typedef struct SNode

{

char data;

struct SNode \* next;

}SNode;

SNode \* Replace(SNode \* x)

{

SNode \* p = x; // 课本定义的链式

while(p != NULL)

{

if(p -> data == 'c')

{

p -> data = 's';

}

p = p -> next;

}

return x;

}

4-21

int pattern\_index(char c[], char s[]) //c为模式串，s为主串

{

int i;

int st; //主串当前下标

int ct; //子串当前下标

for(i = 0; i < StrLen(s); i++)

{

st = i;

ct = 0;

while(st < StrLen(s) && ct < StrLen(c) && (s[st] == c[ct] || s[st] == '?' || c[ct] == '?'))

{

st++;

ct++;

}

if(ct >= StrLen(c))

return i;

}

return -1;

}

4-22

int count(char [s], char c[]) //s为主串，c为子串

{

int i = 0;

int j = 0;

int sum = 0;

while(i < StrLen(s) && j < StrLen(c))

{

if(s[i] == c[j])

{

i++;

j++;

}

else

{

i = i - j + 1;

j = 0;

}

if(j == StrLen(c))

{

j = 0;

sum++;

}

}

return sum;

}

4-23

char Different(SNode \* s, SNode \* t)

{

char retval = '\0';

SNode \* ps = s;

SNode \* pt = t;

while(ps)

{

while(pt)

{

retval = ps -> data;

if(ps -> data == pt -> data)

{

retval = '\0';

break;

}

pt = pt -> next;

}

if(retval != '\0')

break;

ps = ps -> next;

}

return retval;

}

4-25

void maandian(int A[M][N])

{

int i;

int j;

bool have = false;

int min\_row[M];

int max\_col[N];

//将每行最小的元素存起来

for(i = 0; i < M; ++i)

{

min\_row[i] = A[i][0];

for(j = 0; j < N; ++j)

{

if(A[i][j] < min\_row[i])

{

min\_row[i] = A[i][j];

}

}

}

//将每列最大的元素存起来

for(j = 0; j < N; ++j)

{

max\_col[j] = A[0][j];

for(i = 0; i < M; ++i)

{

if(A[i][j] > max\_col[j])

{

max\_col[j] = A[i][j];

}

}

}

//将每个min\_row中的元素都与max\_col中所有元素比较，若有相等，则为马鞍点

for(i = 0; i < M; ++i)

{

for(j = 0; j < N; ++j)

{

if(min\_row[i] == max\_col[j])

{

printf("A[%d][%d] = %d\n", i, j, A[i][j]);

have = true;

}

}

}

if(have == false)

printf("No maandian\n");

}

4-26

typedef int datatype;

typedef struct

{

int i, j; //行号，列好

datatype v; //元素值

}node;

typedef struct

{

int m, n, t; //行数，列数，非零元素个数

node data[SMAX];

}spmatrix;

bool matrix\_mul(spmatrix A, spmatrix B, spmatrix &C)

{

int v = 0; //C中非零元素个数

if(A.n != B.m)

return false; //A的列数如果不等于B的行数，矩阵无法相乘

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

{

for(int l = 0; l < B.n; ++l)

{

int s = 0;

for(int p = 0; p < A.n; p++)

{

s += get\_value(A, k, p) \* get\_value(B, p, l);

}

if(s != 0)

{

C.data[v].m = k;

C.data[v].n = l;

C.data[v].value = s;

v++;

}

}

}

C.m = A.m;

C.n = A.n;

C.t = v;

return true;

}

2-27

void matrix\_mul(int a[], int b[], int c[][], int n)

{

int i, j, k, i1, i2, s;

for(i = 0; i < n; ++i)

{

for(j = 0; j < n; ++j)

{

s = 0;

for(k = 0; k < n; ++k)

{

if(i >= k)

{

i1 = n \* k - (k\*k + k)/2 + i;

}

else

i1 = n \* i - (i\*i + i)/2 + k;

if(k >= j)

{

i2 = n \* j - (j\*j + j)/2 + k;

}

else

{

i2 = n \* k - (k\*k + k)/2 + j;

}

s += a[i1] \* b[i2];

}

c[i][j] = s;

}

}

}