 实 验 报 告

课程名称： 数据结构（C++版）

题 目： 对称矩阵的压缩存储

专业班级： 18空间

姓 名： 邓珊珊

学 号： 1812001107

指导老师： 袁莹

实验时间： 2019.11.21

第一次实验报告

班级： 18空间 姓名：邓珊珊 学号：1812001107

1. 实验目的及原理

（选择公式的原理，及文字说明，所选的公式需要输入什么，计算出什么，如何计算，目的：通过实际问题学会用函数编程实现）

1. 实验目的
2. 掌握对称矩阵的压缩存储方法；
3. 验证对称矩阵压缩存储的寻址方法。
4. 实验内容
5. 建立一个n X n的对称矩阵A;
6. 将对称矩阵用一维数组SA存储；
7. 在数组SA中实现对矩阵A 的任意元素进行存取操作。
8. 实现提示

首先建立一个n X n的对称矩阵A并初始化矩阵的元素。对称矩阵只需存储下三角部分，即将一个n X n的对称矩阵用一个大小为n X （n+1）/2的一维数组SA来存储，则下三角中的元素aij(i>=j)在SA中的下标k与i、j的关系为k=j X (i-1)/2+j.

1. 实验内容

（代码）

#include <iostream>

using namespace std;

template<class T>

class SymmetricMatrix

{

public:

SymmetricMatrix(T\* array, size\_t n)

{

\_arraySize = n\*(n + 1) / 2;

\_size = n;

\_array = new T[\_arraySize];

assert(array);

for (size\_t i = 0; i < n; i++)

{

for (size\_t j = 0; j < n; j++)

{

\_array[i\*(i + 1) / 2 + j] = array[i\*n + j];

}

}

}

T& GetPos(size\_t row, size\_t col) // 获取节点

{

//如果该位置为上三角的，利用对称原理，交换该位置的行和列即可

if (row < col)

{

swap(row, col);

}

return \_array[row\*(row + 1) / 2 + col];

}

void Display() //打印

{

for (int i = 0; i < \_size; i++)

{

for (int j = 0; j < \_size; j++)

{

if (i >= j)

{

cout << \_array[i\*(i + 1) / 2 + j] << " ";

}

else if (i<j)

{

cout << \_array[j\*(j + 1) / 2 + i] << " ";

}

}

cout << endl;

}

cout << endl;

}

private:

T \*\_array; //压缩矩阵

size\_t \_size; //方阵大小\_size\*\_size

size\_t \_arraySize; //压缩矩阵的总大小

};

const int N=5;

int main()

{

int A[N][N],SA[N \* (N+1)/2]={0};

int i,j;

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

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

A[i][j] = A[j][i] = i+j;

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

{

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

cout<<A[i][j]<<" ";

cout<<endl;

}

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

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

SA[i \* (i-1)/2+j] = A[i][j];

SA[i\* (i -1)/2 + j] = A[i][j];

cout<<"请输入行号和列号:";

cin>>i>>j;

cout<<i<<"行"<<j<<"列的元素值是：";

if(i>=j)

cout<<SA[i \* (i - 1)/2 + j]<<endl;

else

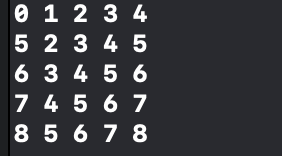
cout<<SA[j\*(j-1)/2 + i]<<endl;

return 0;

}

1. 实验成果

（运行结果截图）



1. 实验心得

通过实验学习，了解了三角矩阵的压缩存储的算法，了解到像下三角矩阵这样的规则矩阵可以压缩存储，极大节省了空间，是我受益匪浅。