P163 32

下三角矩阵和上三角矩阵相乘函数

方法时间复杂度为 O(n^3)

```
template <class T>
inline T ** lowerTriangularMatrix<T>::operator*(const
higherTriangularMatrix<T>&x)
{
    if(this->size()!=x.size())
        throw("dismatch matrix");
    int n=size();
    //创建二维数组
    T ** w=new T*[n];
    for(int i=0;i<n;i++)
        w[i]=new T[n];
    //二维数组赋值
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=n;j++)
        {
            T sum=0;
            for(int k=1;k<=n;k++)
            {
                sum+=get(i,k)*x.get(k,j);
            }
            w[i-1][j-1]=sum;
        }
    }
    return w;
}
```

下三角矩阵的 hpp 文件

```
#include"higherTriangularMatrix.hpp"

//下三角矩阵
template<class T>
class lowerTriangularMatrix
{
public:
    //构造析构函数
    lowerTriangularMatrix(int theN=10);
    lowerTriangularMatrix(const lowerTriangularMatrix<T>&s);
    ~lowerTriangularMatrix(){delete[]element;}
    int size()const {return n;}
    void set(int i,int j,const T& value);
    T get(int i,int j)const;
```

```
    T** operator*(const higherTriangularMatrix<T>&x);
    void output(ostream& out);
private:
    int n;
    T* element;
};

template <class T>
inline lowerTriangularMatrix<T>::lowerTriangularMatrix(int theN)
{
    if(theN<1)
        throw ("matrix size must be >0");
    n=theN;
    element=new T[n*(n+1)/2];
}

template <class T>
inline void lowerTriangularMatrix<T>::set(int i, int j, const T &value)
{
     if(i<1||i>n||j<1||j>n)
        throw("index out of matrix");
    if(i>=j)
    {
        element[i*(i-1)/2+j-1]=value;
        //按列映射则
        //element[(2n-(j-2))*(j-1)/2+i-j]=value;
    }
    else
    {
        if(value!=0)
            throw("elements not in lower triangle must be zero");
    }
}
template <class T>
inline lowerTriangularMatrix<T>::lowerTriangularMatrix(const
lowerTriangularMatrix<T> &s)
{
    this->n=s.n;
    int len=n*(n+1)/2;
    element=new T[len];
    copy(s.element,s.element+len,element);
}

template <class T>
```

```
inline T lowerTriangularMatrix<T>::get(int i, int j) const
{
    if(i<1||i>n||j<1||j>n)
        throw("index out of matrix");
    if(i>=j)
        return element[i*(i-1)/2+j-1];
    else
        return 0;
}
template <class T>
inline void lowerTriangularMatrix<T>::output(ostream& out)
{
    for(int i=1;i<=size();i++)
    {
        for(int j=1;j<=size();j++)
            out<<get(i,j)<<" ";
        out<<endl;
    }
}
```

上三角矩阵的 hpp 文件

```
//下三角矩阵
#include<iostream>
using namespace std;
template<class T>
class higherTriangularMatrix
{
public:
    //构造析构函数
    higherTriangularMatrix(int theN=10);
    higherTriangularMatrix(const higherTriangularMatrix<T>& s);
    ~higherTriangularMatrix(){delete[]element;}
    int size()const {return n;}
    void set(int i,int j,const T& value);
    T get(int i,int j)const;
    void output(ostream& out);
private:
    int n;
    T* element;
};

template <class T>
inline higherTriangularMatrix<T>::higherTriangularMatrix(int theN)
{
```

```
    if(theN<1)
        throw ("matrix size must be >0");
    n=theN;
    //1+2+...+n
    element=new T[n*(n+1)/2];
}

//拷贝构造很重要
template <class T>
inline higherTriangularMatrix<T>::higherTriangularMatrix(const
higherTriangularMatrix<T> &s)
{
    this->n=s.n;
    int len=n*(n+1)/2;
    element=new T[len];
    copy(s.element,s.element+len,element);
}

template <class T>
inline void higherTriangularMatrix<T>::set(int i, int j, const T &value)
{
     if(i<1||i>n||j<1||j>n)
        throw("index out of matrix");
    if(i<=j)
    {
        element[(2*n-(i-2))*(i-1)/2+j-i]=value;
    }
    else
    {
        if(value!=0)
            throw("elements not in lower triangle must be zero");
    }
}

template <class T>
inline T higherTriangularMatrix<T>::get(int i, int j) const
{
    if(i<1||i>n||j<1||j>n)
        throw("index out of matrix");
    if(i<=j)
        return element[(2*n-i+2)*(i-1)/2+j-i];
    else
        return 0;
}
```

```
template <class T>
inline void higherTriangularMatrix<T>::output(ostream& out)
{
    for(int i=1;i<=size();i++)
    {
        for(int j=1;j<=size();j++)
            out<<get(i,j)<<" ";
        out<<endl;
    }
}
```

测试案例

```
#include"lowerTriangularMatrix.hpp"
template<class T>
ostream& operator<<(ostream& out,lowerTriangularMatrix<T>x)
{
   x.output(out);
    return out;
}

template<class T>
ostream& operator<<(ostream& out,higherTriangularMatrix<T>x)
{
    x.output(out);
    return out;
}

int main()
{
    lowerTriangularMatrix<int>a(4);
    a.set(1,1,1);
    a.set(2,1,1);a.set(2,2,1);
    a.set(3,1,1);a.set(3,2,1);a.set(3,3,1);
    a.set(4,1,1);a.set(4,2,1);a.set(4,3,1);a.set(4,4,1);
    cout<<"matrix a"<<endl;
    cout<<a;
    higherTriangularMatrix<int>b(4);
    b.set(1,1,2);b.set(1,2,2);b.set(1,3,2);b.set(1,4,2);
    b.set(2,2,2);b.set(2,3,2);b.set(2,4,2);
    b.set(3,3,2);b.set(3,4,2);
    b.set(4,4,2);
    cout<<"matrix b"<<endl;
    cout<<b;
```

```
    int n=a.size();
    int ** c=a*b;

    cout<<"matrix a*b"<<endl;
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            cout<<c[i][j]<<" ";
        cout<<endl;
    }
    return 0;
}
```

输出

```
matrix a
1 0 0 0
1 1 0 0
1 1 1 0
1 1 1 1
matrix b
2 2 2 2
0 2 2 2
0 0 2 2
0 0 0 2
matrix a*b
2 2 2 2
2 4 4 4
2 4 6 6
2 4 6 8
```

P173 T41

get 函数

时间复杂度 O(terms.size())

```
template <class T>
inline T SparseMatrix<T>::get(int theRow, int theColumn)
{
    for(int i=0;i<terms.size();i++)
    {
        auto k = terms.get(i);
        if (k.row == theRow && k.col == theColumn)
            return k.value;
    }
    throw("no element");
```

```
}
```

set 函数

时间复杂度 O(1)

```
template<class T>
inline void SparseMatrix<T>::set(int theRow, int theColumn, int
theValue)
{
    if(terms.size()==terms.capacity())
        terms.reserve(terms.size()*2);
    //赋值
    MatrixTerms<T>mTerm;
    mTerm.row=theRow;
    mTerm.col=theColumn;
    mTerm.value=theValue;

    terms.push_back(mTerm);
}
```

测试案例

```
#include"spareMatrix.hpp"
int main()
{
    SparseMatrix<int>s(3,3,3);
    s.set(2,2,1);
    s.set(2,3,2);
    s.set(3,1,1);
    cout<<s;
    cout<<s.get(2,2)<<endl;
    cout<<s.get(2,3)<<endl;
    cout<<s.get(3,1);
    return 0;
}
```

输出

```
rows=3 columns=3
nonZeroTerms=3
a(2,2)=1
a(2,3)=2
a(3,1)=1
1
2
1
```