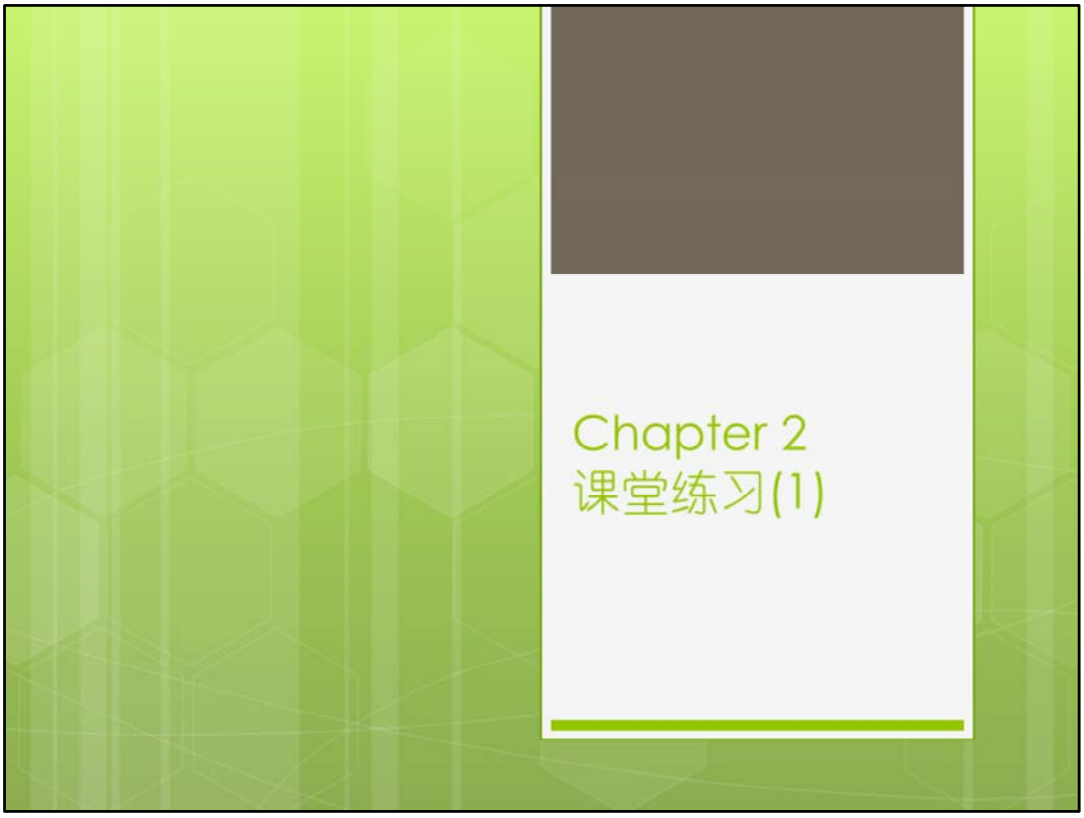
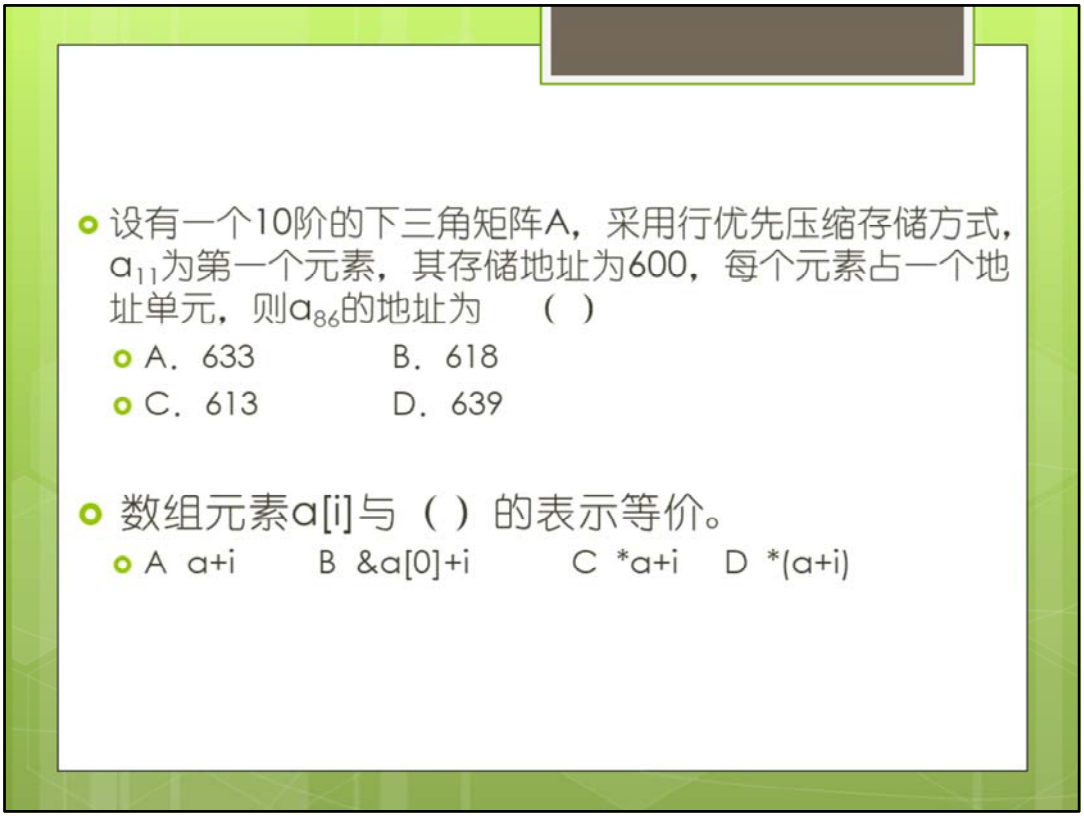
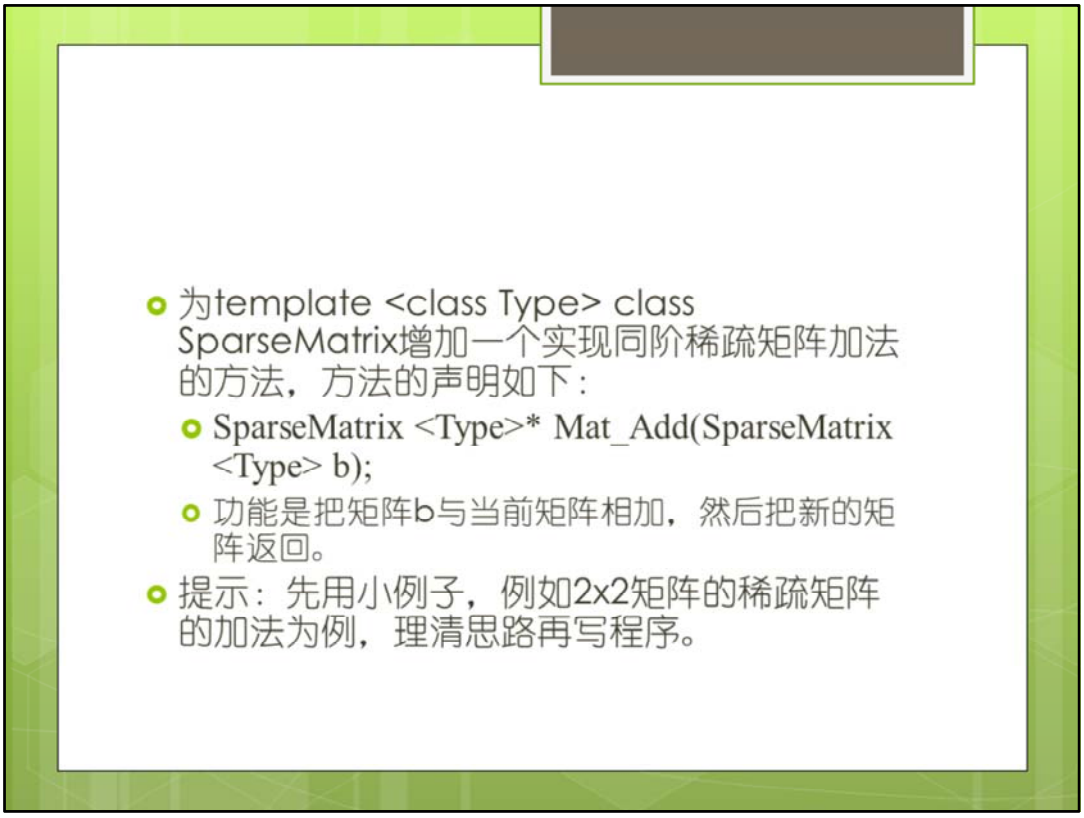

Chapter 2
课堂练习(1)



答案：

（1）600+（1+7）/2*7+5=633，所以答案为A

（2）D




```
#include <iostream>

using namespace std;

#define MaxTerms 100

template<class Type> class SparseMatrix;
//稀疏矩阵类的前向引用声明
template <class Type> class Trituple
{  //三元组类定义
      friend class SparseMatrix <Type>;
public:
       Trituple(){};
      void Set(int r,int c,Type d){
                                          row = r;
                                          col = c;
                                          data = d;
      }

private:
      int row, col;  //非零元素的行号与列号
      Type data;  //非零元素的值
};

template <class Type> class SparseMatrix {
//稀疏矩阵类定义
```

```
   public:
      SparseMatrix(int MaxRow, int MaxCol,Trituple <Type> sm[MaxTerms],int nzt);
      //构造函数：建立一个MaxRow行与MaxCol列的稀疏矩阵
      SparseMatrix <Type>* Mat_Add(SparseMatrix <Type> b);
      void Mat_out();

    private:
       int Rows, Cols, NonZero_Terms;
       Trituple <Type> SMArray[MaxTerms];
};

template <class Type> void SparseMatrix <Type> :: Mat_out ()
{
                for(int i=0;i<NonZero_Terms;i++) cout << endl <<SMArray[i].row <<
SMArray[i].col << SMArray[i].data;
}

template <class Type> SparseMatrix <Type> :: SparseMatrix (int MaxRow, int MaxCol,Trituple
<Type> sm[MaxTerms],int nzt)
{
                for(int i=0;i<nzt;i++) SMArray[i]=sm[i];
                Rows = MaxRow;
                Cols = MaxCol;
                NonZero_Terms = nzt;
}

template <class Type> SparseMatrix <Type>* SparseMatrix <Type> :: Mat_Add(SparseMatrix
<Type> b)
{
                SparseMatrix <Type> *c=new SparseMatrix <Type>(Rows,Cols,SMArray,0);
                int i=0, j=0,k=0;

                while((i<NonZero_Terms)&&(j<b.NonZero_Terms)){

                if((SMArray[i].row<b.SMArray[j].row)||((SMArray[i].row==b.SMArray[j].row)&
&(SMArray[i].col<b.SMArray[j].col))) {
                                        c->SMArray[k].row = SMArray[i].row;
                                        c->SMArray[k].col = SMArray[i].col;
                                        c->SMArray[k].data = SMArray[i].data;
                                        k++;
                                        i++;
                                }
                                else
if((SMArray[i].row==b.SMArray[j].row)&&(SMArray[i].col==b.SMArray[j].col)) {
                                        c->SMArray[k].row = SMArray[i].row;
                                        c->SMArray[k].col = SMArray[i].col;
                                        c->SMArray[k].data = SMArray[i].data +
b.SMArray[j].data;
                                        if(c->SMArray[k].data!=0) k++;
                                        i++;
                                        j++;
                                }
```

```
                        else{
                                c->SMArray[k].row = b.SMArray[j].row;
                                c->SMArray[k].col = b.SMArray[j].col;
                                c->SMArray[k].data = b.SMArray[j].data;
                                k++;
                                j++;
                        }
                }
                while(i<NonZero_Terms){
                        c->SMArray[k].row = SMArray[i].row;
                        c->SMArray[k].col = SMArray[i].col;
                        c->SMArray[k].data = SMArray[i].data;
                        k++;
                        i++;
                }
                while(j<b.NonZero_Terms){
                        c->SMArray[k].row = SMArray[j].row;
                        c->SMArray[k].col = SMArray[j].col;
                        c->SMArray[k].data = SMArray[j].data;
                        k++;
                        j++;
                }

                c->NonZero_Terms = k;

   return c;
}

int main(int argc, char* argv[]) {

                Trituple <int> x[3], y[3];
                x[0].Set(1,2,3);
                x[1].Set(2,2,1);
                x[2].Set(3,3,1);

                y[0].Set(2,2,7);
                y[1].Set(2,3,3);
                y[2].Set(3,3,9);

                SparseMatrix <int> a(10,20,x,3),b(10,20,y,3),*c;

                c=a.Mat_Add(b);

                c->Mat_out ();

                return 0;

}
```