



- 设有一个10阶的下三角矩阵A，采用行优先压缩存储方式， $a_{11}$ 为第一个元素，其存储地址为600，每个元素占一个地址单元，则 $a_{86}$ 的地址为 ( )
  - A. 633                      B. 618
  - C. 613                      D. 639
- 数组元素 $a[i]$ 与 ( ) 的表示等价。
  - A  $a+i$       B  $\&a[0]+i$       C  $*a+i$       D  $*(a+i)$

答案：

(1)  $600 + (1+7) / 2 * 7 + 5 = 633$ ，所以答案为A

(2) D

- 为template <class Type> class SparseMatrix增加一个实现同阶稀疏矩阵加法的方法，方法的声明如下：
  - SparseMatrix <Type>\* Mat\_Add(SparseMatrix <Type> b);
  - 功能是把矩阵b与当前矩阵相加，然后把新的矩阵返回。
  - 提示：先用小例子，例如2x2矩阵的稀疏矩阵的加法为例，理清思路再写程序。

```
#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> SArray[MaxTerms];
};

template <class Type> void SparseMatrix <Type> :: Mat_out ()
{
    for(int i=0; i<NonZero_Terms; i++) cout << endl << SArray[i].row <<
    SArray[i].col << SArray[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++) SArray[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, SArray, 0);
    int i=0, j=0, k=0;

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

        if((SArray[i].row<b.SArray[j].row)||((SArray[i].row==b.SArray[j].ro
w)&&(SArray[i].col<b.SArray[j].col))) {
            c->SArray[k].row = SArray[i].row;
            c->SArray[k].col = SArray[i].col;
            c->SArray[k].data = SArray[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;  
  
}
```