

```

// A safe array example.
#include <iostream>
#include <cstdlib>
#include <string.h>
using namespace std;

class atype{
    int ncols;
    int nrows;
    int **dynamicArray;

    public:
    atype(){
        nrows=0;
        ncols=0;
        dynamicArray=0;
    }

    //constructor
    atype(int row, int col){
        nrows=row;
        ncols=col;
        dynamicArray = new int*[nrows];
        for( int i = 0 ; i < nrows ; i++ ){
            dynamicArray[i] = new int [ncols];
        }
    }

    //destructor
    ~atype(){
        if (dynamicArray != 0)
        {
            for (int i=nrows-1; i>=0; i--)
            {
                if (dynamicArray[i] != 0)
                    delete dynamicArray[i];
                dynamicArray[i] = 0;
            }
            delete [] dynamicArray;
        }
        dynamicArray = 0;
    }

    //user inserting elements in 2d array
    void fillArray()
    {
        for (int in=0;in<nrows;in++){
            for (int j=0;j<ncols;j++){
                int value;
                cout<<"enter values";
                cin>>value;
                dynamicArray[in][j] = value;
            }
        }
    }
}

```

```
}  
}
```

```
//bound checking-safe array implementation  
int &operator()(int i, int j){  
    if(i<0 || i> nrows-1 || j<0 || j> ncols-1 ) {  
        cout << "Boundary Error\n";  
        exit(1);  
    }  
    return dynamicArray[i][j];  
}
```

```
//copy constructor  
atype(const atype& rhs)  
{  
    nrows = rhs.nrows;  
    ncols = rhs.ncols;  
    dynamicArray = new int*[nrows];  
    for( int i = 0 ; i < nrows ; i++ ){  
        dynamicArray[i] = new int [ncols];  
        memcpy(dynamicArray[i],rhs.dynamicArray[i],sizeof(int)*ncols);  
    }  
}
```

```
//assignment operator overloading  
atype& operator=(const atype& rhs)  
{  
    if (this == &rhs)  
        return *this;  
    for (int i=nrows-1; i>=0; i--)  
    {  
        delete dynamicArray[i];  
    }  
    delete [] dynamicArray;  
  
    nrows = rhs.nrows;  
    ncols = rhs.ncols;  
    dynamicArray = new int*[nrows];  
    for( int i = 0 ; i < nrows ; i++ ){  
        dynamicArray[i] = new int [ncols];  
        memcpy(dynamicArray[i],rhs.dynamicArray[i],sizeof(int)*ncols);  
    }  
    return *this;  
}
```

```
//not equal to operator overloading  
atype& operator!=(const atype& rhs){  
    for (int i=0;i<nrows;i++){  
        for (int j=0;j<ncols;j++){  
            if(dynamicArray[i][j]!=rhs.dynamicArray[i][j]){  
                cout<<"not equal";  
                break;  
            }  
        }  
    }  
}
```

```
        }  
        break;  
    }  
};
```

```
int main()  
{  
    int columns;  
    int rows;  
    cout<<"enter number of rows and cols"<<endl;  
    cin>>rows>>columns;  
    atype ob1(rows,columns);  
    ob1.fillArray();  
    atype ob2=ob1;  
    atype ob3(3,3);  
    ob3.fillArray();  
    cout << ob1(1,1) << endl;  
    cout<<ob3(1,1)<<endl; //checking bounds of array  
    ob1!=ob3;  
    return 0;  
}
```