“多维数组的设计与实现”

一、设计要求

1、问题描述

尽管 C 和 Pascal 等程序设计语言已经提供了 多维数组，但在某些情况下，定义用户所需的多维数组也是很有用的。通过设计并模拟实现多维数组类型，可以深刻理解和掌握多维数组。

2、需求分析

（1）、定义整型多维数组类型，各维的下标是任意整数开始的连续整数。

（2）、下标变量赋值，执行下标范围检查。

（3）、同类型数组赋值。

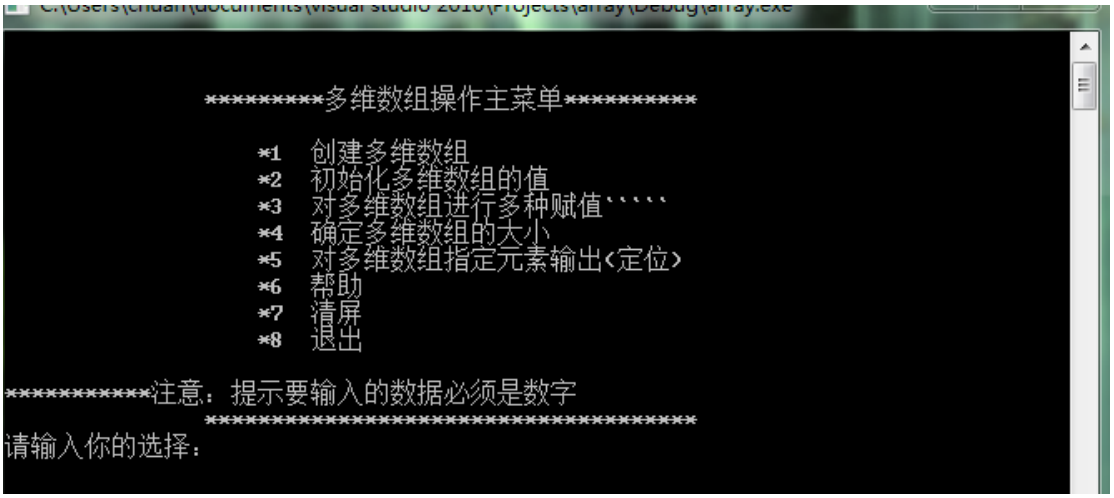
（4）、子数组赋值，例如，a[1```n] = a[2```n+1];a[2```4][3```5] = b[1```3][2``4];

（5）、确定数组的大小。

二、概要设计

1、主界面设计

为了实现多维数组的操作，设计了一个含有多个菜单项的主控菜单子程序以链接系统的各项子功能，方便用户使用本系统。本系统主控菜单运行界面如图下所示：



2、存储结构设计

本系统主要采用动态数组来表示存储多维数组元素值的信息。用结构体变量来保存用于操作多维数组的相关信息。

3、系统功能设计

（1）、创建多维数组

（2）、初始化多维数组的值

（3）、对多维数组进行多种赋值

a、对一个多维数组指定下标的元素赋值

b、对一个多维数组指定子数组进行赋值

c、对一个多维数组同类型的子数组进行赋值

d、对两个多维数组间同类型的子数组进行赋值

e、对两个多维数组间进行赋值

（4）、确定多维数组的大小

（5）、对多维数组指定元素输出(定位)

三、模块设计

1、模块设计

本程序包含两个模块：主程序模块和各个函数模块。

2、系统子程序及功能设计

（1）、open1()　主菜单界面

（2）、open2()　对多维数组赋值子菜单界面

（3）、InitNArray(NArray &A,NArrayType &a)　创建多维数组

（4）、CreatNArray(NArrayType &a,int k)　初始化多维数组的值

（5）、Assign1(NArray &A,NArrayType &a) 对一个多维数组指定下标的元素赋值

（6）、Assign2(NArray &A,NArrayType &a) 对一个多维数组指定子数组进行赋值

（7）、Assign3(NArray &A,NArrayType &a) 对一个多维数组同类型的子数组进行赋值

（8）、Assign4(NArray &A,NArrayType &a,NArray &B,NArrayType &b) 对两个多维数组间同类型的子数组进行赋值

（9）、Assign5(NArray &A,NArrayType &a,NArray &B,NArrayType &b) 对两个多维数组间进行赋值

（10）、Assign(NArray &A,NArrayType &a,NArray &B,NArrayType &b) 对多维数组进行多种赋值

（11）、size(NArray A,NArrayType a) 确定多维数组的大小

（12）、help() 帮助信息

（13）、output(NArray &A,NArrayType &a) 对多维数组指定元素输出(定位)

3、函数主要调用关系图

四、详细设计

1、数据类型定义

```
typedef struct{
    int dim;//数组维数
    int *lower;//各维下表的指针
    int *upper;//各维上界表的指针
    int *constants;//映像函数常量表的指针
} NArray,*NArrayPtr;
typedef struct{
    int *elem;//数组元素基址
    int num;//数组元素个数
} NArrayType;
```

2、系统主要子程序的详细设计

（1）、创建多维数组的函数：

```
void main(){
```

```
int i,j,k,flag;
NArray A,B;
NArrayType a,b;
i = 0;
j = 0;//标志是否创建成功的
k = 1;
flag = 1;
while(k){
    open1();
    printf("请输入你的选择：  ");
    scanf("%d",&i);
    switch(i){
        case 1:{
            InitNArray(A,a);//创建多维数组
            j = 1;
            break;
              }
        case 2:{
            if(j){
                CreatNArray(a,1);//前提要先创建成功，初始化多维数组的值

                break;
            }else {
                printf("你还没有创建数组，按任意键返回，创建数组：\n");
                getch();
                break;
            }
             }
        case 3:{//前提要创建成功，对多维数组进行多种赋值`````
            if(j){
                if(flag){
                    printf("请先创建，初始化多维数组  B\n,按任意键继续
\n");

                    getch();
                    InitNArray(B,b);
                    CreatNArray(b,0);
                    flag = 0;
                }

                Assign(A,a,B,b);
                break;
            }else {
                printf("你还没有创建数组，按任意键返回，创建数组：\n");
                getch();
```

```
                    break;
                }
                }
            case 4:{
                if(j){//前提要创建成功，确定多维数组的大小
                    size(A,a);
                    break;
                }else {
                    printf("你还没有创建数组，按任意键返回，创建数组：\n");
                    getch();
                    break;
                }
                }
            case 5:{
                if(j){//前提要创建成功，对多维数组指定元素输出(定位)
                    output(A,a);
                    break;
                }else{
                    printf("你还没有创建数组，按任意键返回，创建数组：\n");
                    getch();
                    break;
                }
                }
            case 6:{
                help();//帮助
                break;
                }
            case 7:{
                system("cls");//清屏
                break;
                }
            case 8:{
                k=0;
                break;//退出
                }
            default:
                printf("\n 对不起!你的输入结果不正确!请重新输入!\n");
            }

        }
    }
```

void InitNArray(NArray &A,NArrayType &a){//定义一个多维数组，各维下标是任意整数开始的连续整数，手动输入

```
        printf("                    *********创建一个多维数组*********\n");
        printf("请按提示创建：\n");
        int i,u,l,n,w;
        int flag1 ,flag2,flag3;
        flag1 = 1;
        while(flag1){
            printf("输入你想要的不大于%d 的多维数组的维数，输入维数：
",MAXDIM);
            scanf("%d", &w);
            if(w<1 || w>MAXDIM) {
                printf("维数不在处理范围内，按任意键返回，重新输入: \n");
                getch();
            }else {
                A.dim = w;//确定多维数组的维数，手动输入确定
                flag1 = 0;
            }
        }
        A.lower = (int *)malloc(A.dim * sizeof(int));//为各维的下界数值开拓存储空间
        A.upper = (int *)malloc(A.dim * sizeof(int));//为各维上界数值开拓存储空间
        for(i = 0;i<A.dim;i++){
            A.lower[i] = 0;
            A.upper[i] = 0;//初始化各维上下界数值值为 0
        }
        for(i = 0;i<A.dim; i++){
            flag1 = 1;
            while(flag1){
                flag2 = 1;
                flag3 = 1;
                while(flag2){
                    printf("输入第%d 维的上界: ",i+1);
                    scanf("%d",&u);
                    if(u<0){
                        printf("输入的数据不在处理范围内，按任意键返回，重新输
入,：\n");//上界数值不能为 0
                        getch();
                    }else{
                        A.upper[i] = u;//用 A.upper[]数组存储各维上界数值
                        flag2 = 0;
                    }
                }
                while(flag3){
                    printf("输入第%d 维的下界：  ",i+1);
                    scanf("%d",&l);
                    if(l<0){
```

```
                    printf("输入的数据不在处理范围内，按任意键返回，重新输入：\n");//下界数值不能为 0
                    getch();
                }else {
                    A.lower[i] = l;//用 A.lower[]数组存储各维下界数值
                    flag3 = 0;
                }
            }
            if(A.lower[i]<=A.upper[i]){//判断下界数值是不是小于上界数值
                flag1 = 0;
            }else {
                printf("你输入的上界大于下界，按任意键返回，重新输入：\n");
                getch();
            }
        }

    }
    A.constants = (int *)malloc(A.dim * sizeof(int));//为各维映像函数常量表开拓存储空间
    A.constants[A.dim-1] = 1;
    for(i = A.dim-2;i>=0;i--){
        A.constants[i] = A.constants[i+1] *(A.upper[i+1] - A.lower[i+1]+1);//按公式初始化各维映像函数常量表数值，
    }
    n=1;
    for(i = 0;i<A.dim;i++){
        n = n*(A.upper[i] - A.lower[i] +1);//计算所输入的多维数组的大小
    }
    a.num = n;
    a.elem = (int *)malloc(a.num * sizeof(int));//根据确定的多维数组大小来动态开拓存储空间
    for(i = 0;i<a.num;i++)
        a.elem[i] = 0;//设初始化多维数组的值为 0
    printf("你已经成功创建了一个%d 维的，大小是%d，多维数组 A",A.dim,a.num);
    for(i=0 ;i<A.dim; i++){
        printf("[%d```%d]",A.lower[i],A.upper[i]);
    }
    printf("\n 按任意键继续：\n");
    getch();

}
```

（2）、对两个多维数组间同类型的子数组进行赋值

```
void Assign4(NArray &A,NArrayType &a,NArray &B,NArrayType &b){
    printf("                    **********对两个多维数组间同类型的子数组组间
```

```
进行赋值*********\n");
    printf("请按提示进行赋值：\n");
    int i,j,m1[MAXDIM],m2[MAXDIM],n1[MAXDIM],n2[MAXDIM];
    int flag1,flag2;
    int x,y;
    int num1,num2;
    flag2 = 1;
    while(flag2){
        for(i=0; i<A.dim; i++){
            flag1 =1;
            while(flag1){
                printf("对于%d 维数组 A，输入你想替换的子数组 A1，第%d 维起始元素下标值，范围是[%d```%d]: ",A.dim,i+1,A.lower[i],A.upper[i]);
                scanf("%d",&x);
                if(x<=A.upper[i] && x>=A.lower[i]){
                    m1[i] = x;//用 m1[]数组存储 A 数组的子数组起始元素下标，存储单元依次存取按顺序各维的下标
                    flag1 = 0;
                }else{
                    printf("你输入的下标不在处理的范围内，按任意键返回，重新输入：\n");
                    getch();//出错处理，各维元素的下标值不能大于各维上界，不能小于各维下界，不然重新输入
                }
            }
        }
        for(i=0; i<A.dim; i++){
            flag1 = 1;
            while(flag1){
                printf("对于%d 维数组 A，输入你想替换的子数组 A1，第%d 维结束元素下标值，范围是[%d```%d]: ",A.dim,i+1,A.lower[i],A.upper[i]);
                scanf("%d",&y);
                if(x<=A.upper[i] && x>=A.lower[i]){
                    m2[i] = y;//用 m2[]数组存储 A 数组的子数组结束元素下标，存储单元依次存取按顺序各维的下标
                    flag1 = 0;
                }else{
                    printf("你输入的下标不在处理的范围内，按任意键返回，重新输入：\n");
                    getch();//出错处理，各维元素的下标值不能大于各维上界，不能小于各维下界，不然重新输入
                }
            }
        }
```

```
    for(i=0; i<B.dim; i++){
        flag1 =1;
        while(flag1){
            printf("对于%d 维数组 B，输入你想替换的子数组 B1，第%d 维起始元素下标值，范围是[%d```%d]: ",A.dim,i+1,A.lower[i],A.upper[i]);
            scanf("%d",&x);
            if(x<=B.upper[i] && x>=B.lower[i]){
                n1[i] = x;//用 n1[]数组存储 A 数组的子数组结束元素下标，存储单元依次存取按顺序各维的下标
                flag1 = 0;
            }else{
                printf("你输入的下标不在处理的范围内，按任意键返回，重新输入：\n");
                getch();//出错处理，各维元素的下标值不能大于各维上界，不能小于各维下界，不然重新输入
            }
        }
    }
    for(i=0; i<B.dim; i++){
        flag1 = 1;
        while(flag1){
            printf("对于%d 维数组 B，输入你想替换的子数组 B1，第%d 维结束元素下标值，范围是[%d```%d]: ",A.dim,i+1,A.lower[i],A.upper[i]);
            scanf("%d",&y);
            if(x<=B.upper[i] && x>=B.lower[i]){
                n2[i] = y;//用 n2[]数组存储 A 数组的子数组结束元素下标，存储单元依次存取按顺序各维的下标
                flag1 = 0;
            }else{
                printf("你输入的下标不在处理的范围内，按任意键返回，重新输入：\n");
                getch();//出错处理，各维元素的下标值不能大于各维上界，不能小于各维下界，不然重新输入
            }
        }
    }
    num1 = 1;
    num2 = 1;
    for(i=0; i<A.dim; i++){
        num1 = num1 * (abs(m2[i] - m1[i]) +1);//计算所选子 A 数组的子数组 A1 的大小
        num2 = num2 * (abs(n2[i] - n1[i]) +1);//计算所选子 B 数组的子数组 B1 的大小
    }
```

```
        if(num1 == num2)
            flag2 = 0;
        else
            printf("两个子数组不是同种类型的，按任意键返回，重新输入： \n");
            getch();
    }
    printf("\n");
    printf("你选择多维数组 A 的子数组 A 是从： A1");
    for(i=0; i<A.dim; i++){
        printf("[%d]",m1[i]);
    }
    printf("到~~A1");
    for(i=0; i<A.dim;i++){
        printf("[%d]",m2[i]);
    }
    printf("你选择的子数组 B 是从： B1");
    for(i=0; i<B.dim; i++){
        printf("[%d]",n1[i]);
    }
    printf("到~~B1");
    for(i=0; i<B.dim;i++){
        printf("[%d]",n2[i]);
    }
    printf("\n");
    int posA1,posA2,posB1,posB2;
    posA1 =0;
    posA2 = 0;
    posB1 =0;
    posB2 = 0;
    for(i=0 ;i<A.dim; i++){
        posA1 = posA1 + A.constants[i] * (m1[i] - A.lower[i]);//根据下标值用公式计算 A 数组的子数组起始地址所对应一维数组的位置
        posA2 = posA2 + A.constants[i] * (m2[i] - A.lower[i]);//根据下标值用公式计算 A 数组的子数组起始地址所对应一维数组的位置
        posB1 = posB1 + A.constants[i] * (n1[i] - A.lower[i]);//根据下标值用公式计算 A 数组的子数组起始地址所对应一维数组的位置
        posB2 =posB2 + A.constants[i] * (n2[i] - A.lower[i]);//根据下标值用公式计算 A 数组的子数组起始地址所对应一维数组的位置
    }
    printf("你选的子数组的值是： ");
    for(i = posA1; i<=posA2; i++){
        printf("%d  ",a.elem[i]);
    }
    for(i = posA1,j = posB1; i<=posA2 && j<=posB2; i++,j++){
```

```
            a.elem[i] = b.elem[j];//把 B 数组的子数组 B1 的值赋给 A 数组的子数组 A1
        }
        printf("\n 赋值后，多维数组的值是：\n");
        for(i=posA1; i<=posA2; i++){
            printf("%d  ",a.elem[i]);//输出赋值后的 A 数组的子数组 A1 的值
        }
        printf("\n 赋值成功，按任意键继续：\n");
        getch();

    }
```

五、测试分析

1、创建多维数组

操作步骤：

a、在主操作菜单下，用户输入“1”并回车，创建一个多维数组。

测试：现在创建一个 2 维数组 A[2```5][3```4],大小为 8，并把它的全部元素值初始化为 0；即是：A]2``5][3``4]={0,0,0,0,0,0,0,0}，

异常输入处理：输入的维数大于 5（开始时最大维数已经设置好了），会提示你重新输入，不会因此而崩溃。

2、初始化多维数组的值

初始化要在创建了多维数组的前提下进行，现在我将创建好的多维数组，

在主操作菜单下按“2”，就初始化了一个数组。

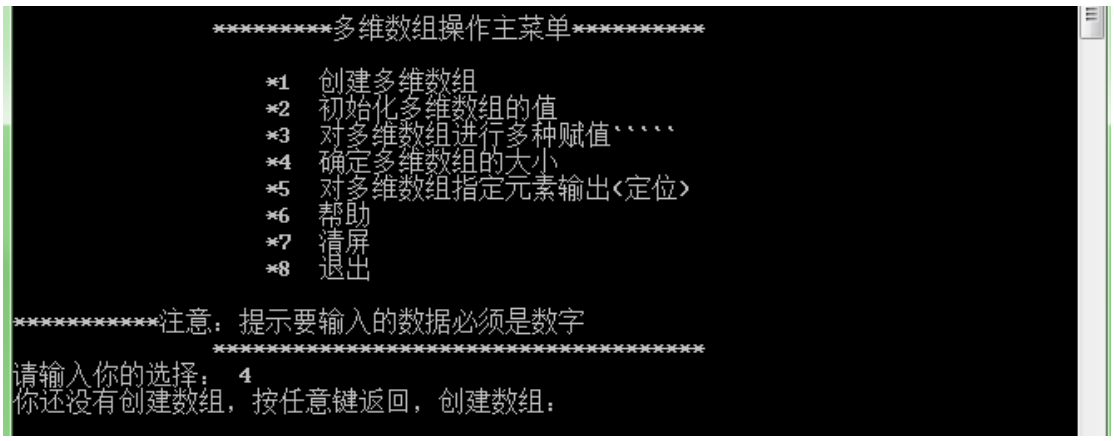
对于 A]2``5][3``4]={0,0,0,0,0,0,0,0,}，

测试：初始化 A]2``5][3``4]={1、2、3、4、5、6、7、8}，初始化的值是我用一个循环一次递增赋值的。

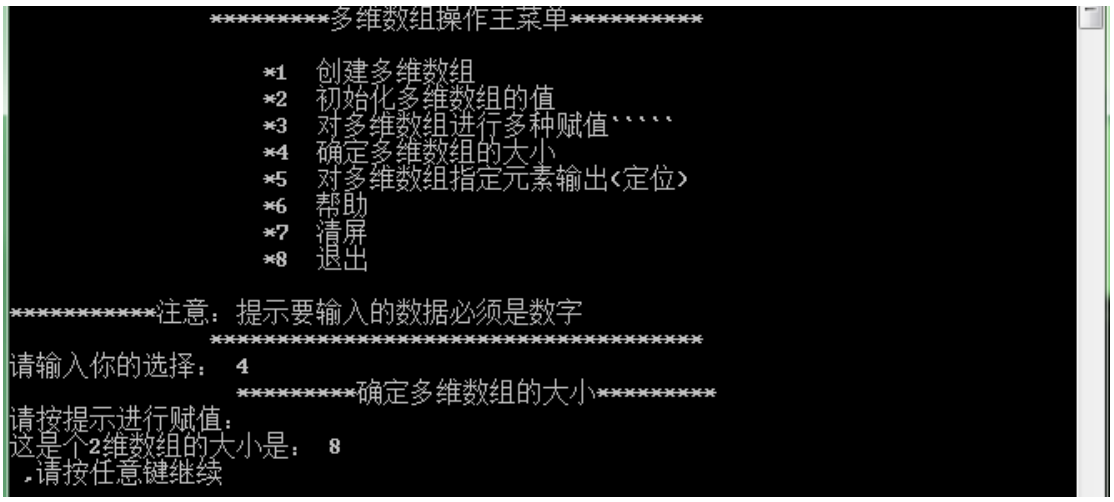
3、确定多维数组的大小

前提也是要在创建多维数组之后才能确定多维数组的大小。

步骤：在主操作菜单上按“4”



测试：对刚才创建的 2 维数组 A]2``5][3``4]={1、2、3、4、5、6、7、8}，可以确定它的大小是 8；



4、对多维数组指定元素输出(定位)

前提也是多维数组要创建。

测试：对刚才创建的 2 维数组 A]2``5][3``4]={1、2、3、4、5、6、7、8}，现在要查找 A[5][3]元素的值，进过查找 A[5][3] = 7;同时输入越界也会有出错处理

5、对一个多维数组指定下标的元素赋值

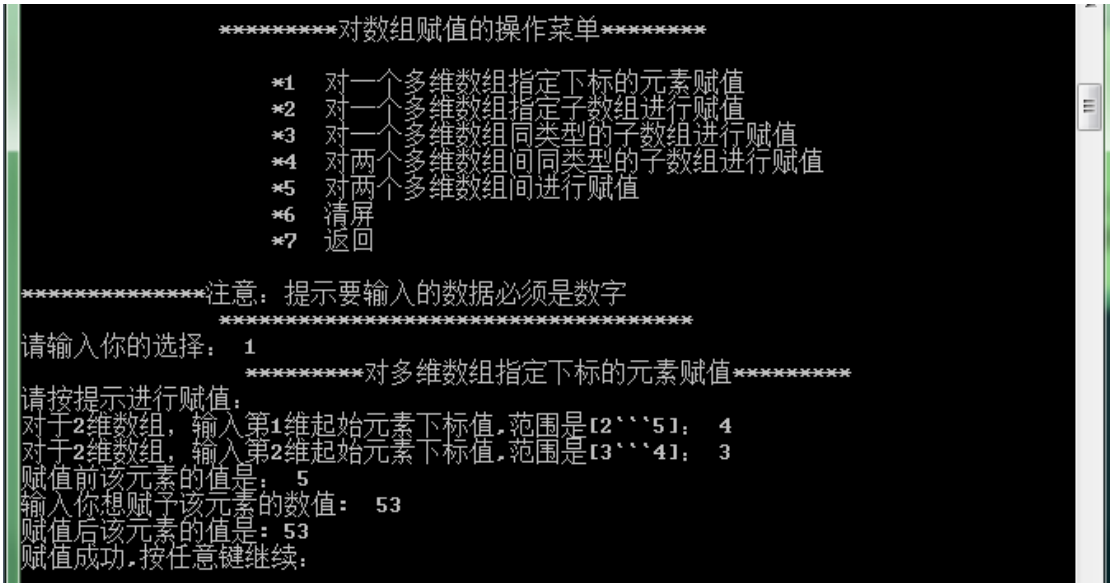
前提也是多维数组已经创建，现在在刚才创建了 A 数组的前提下，修改 A[4][3]的值，并通过人机交互，赋予一个你想要赋予的值。

对于初始化好的多维数组 A]2``5][3``4]={1、2、3、4、5、6、7、8}，

如：测试一：A[4][3]=53;

测试二：A[5][4] =54;

a、在主菜单界面创建按“3” ——>b、接着创建一个 B 数组，因为以后要用到。并自动给 B 数组赋值——>c、接着进入对多维数组赋值操作菜单，然后按“1”——>d、载人机交互如入“53”、——>e、最后输出赋值后的结果



6、对一个多维数组指定子数组进行赋值

前提是多维数组已经创建好。

a、在对多维数组操作菜单下，按“2”——> b、输入数组的起始元素各维下标，再输入数组结束元素各维下标——> c、根据提示手动给各个元素赋值——> d、最后输出赋值后的结果，并与赋值前的比较。

对于多维数组 A]2``5][3``4]={1、2、3、4、5、6、7、8}，

测试一：A[3```4][3```4]的数组赋值,{61,62,63,64}；

赋值前：A[3][3]~~A[4][4]={3,4,5,6,}， 赋值后：A[3][3]~~A[4][4] = {61,62,63,64}；

测试二：A[2```4][4```3]的数组赋值，{41,42,43,44}；

赋值前：A[2][4]~~A[4][3]={2,41,42,43}，赋值后:A[2][4]~~A[4][3] = {41,42,43,44}；

7、对一个多维数组同类型的子数组进行赋值

对于初始化好的多维数组 A[2```5][3```4]={1,2,3,4,5,6,7,8,}

测试一：把子数组 A[3```4][3```4]赋值给 A[2```3][3```4];

赋值前：A[2][3]~~A[3][4]={1,2,3,4},复制后：A[2][3]~~A[3][4]={3,4,5,6,}

测试二：把子数组 A[2```4][3```4]赋值给 A[3```5][3```4];

赋值前：A[3][3]~~A[5][4]={5,6,5,6,7,8},赋值后：A[3][3]~~A[5][4]={3,4,3,4,3,4,}

8、对两个多维数组间同类型的子数组进行赋值

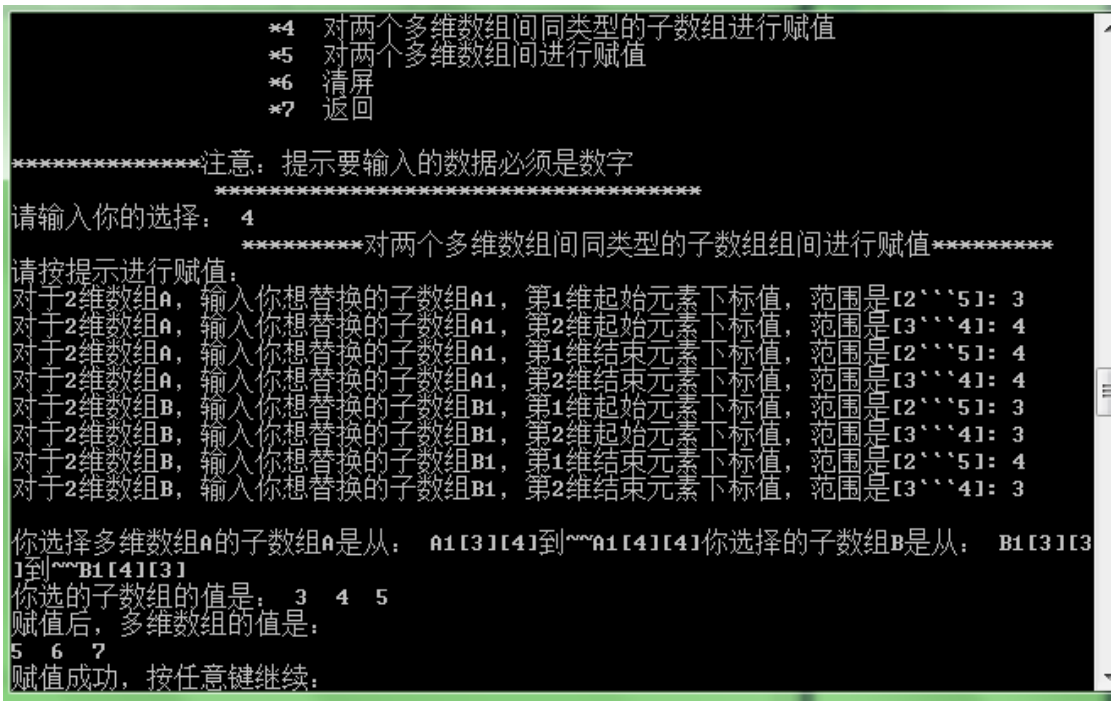
对于已经初始化好的 A[2```5][3```4]={1,2,3,4,5,6,7,8,} ，B[2```5][3```4]={3,4,5,6,7,8,9,10,}，

测试一：把 B[2][3]~~B[4][3]赋值给 A[3][4]~~A[5][4],

赋值前：A[3][4]~~A[5][4]={4,5,6,7,8,}，赋值后 A[3][4]~~A[5][4]={3,4,5,6,7}；

测试二：把 B[3][3]~~B[4][3]赋值给 A[3][4]~~A[4][4],

赋值前：A[3][4]~~A[4][4]={3,4,5} 赋值后：A[3][4]~~A[4][4]={5,6,7}



9、对两个多维数组间进行赋值

测试：对于已经初始化好的 A[2```5][3```4]={1,2,3,4,5,6,7,8,}，B[2```5][3```4]={3,4,5,6,7,8,9,10,}，

赋值前：A[2```5][3```4]={1,2,3,4,5,6,7,8,}，赋值后：A[2```5][3```4]={34,5,6,7,8,9,10}，

六、用户手册

（1）、本程序执行文件为“array.exe”。

（2）、进入本系统之后，随即显示多维数组操作主菜单界面，用户可在该界面下输入各个子菜单前对应的数字并按回车键，执行相应子菜单命令。

（3）、按照提示输入数据，查看数信息。

七、调试分析

设计：

1、先对多维数组的需求进行分析，充分理解所实现的功能

2、对实现多维数组的操作有个总体的设计，并用草稿纸画出他的层次图。

3、定好所用的数据存储类型。

4、确定好实现功能所需要的函数，并画图理清函数之间的层次关系与调用关系。

5、写每个子函数，实现各个功能，完成一个功能就测试，以便以后整合后排查错误困难。

6、写 main（）函数，整合所有的子函数，实现功能的整合。

7、界面设计，尽量做到人机交互简单，容易理解。

8、测试分析。

9、写实验报告，总结，反思，分享，讨论。

调试：

测试函数时会遇到很多边界的错误，数组指针越界，参数没有初始化参与运算，造成了错误。

1、我很多时候利用断点，单步调试，先预先判定在哪个循环出错可能性比较大，然后在那个循环开始前设置断定，逐过程进行调试，并查看参数的变化，不过 VS2010 不怎么会用，不会过滤掉一些参数的信息，去看你想要的参数的变化，造成了调试的困难。发现很多细节出错，“<=”写成“<”,粗心。

2、有些事编译出错，也是粗心，根据 VS 的提示可以一一排查。

八：附录：

源程序文件名清单：

function.h//各个功能的实现函数

main1.cpp//main 函数

store.h//多维数组的存储结构