****

**实 验 报 告**

**实验课程： 数据结构与算法实验**

**实验题目： 二叉树线索化相关操作**

**任课老师： 吴之旭老师**

**学生姓名： 殷骢睿**

**学 号： 5509121041 \_**

**专业班级： 2021级人工智能实验班 \_**

**2022 年 11 月 26日**

** 南昌大学实验报告**

学生姓名： 殷骢睿 学号： 5509121041 专业班级： 人工智能211班

实验类型：□ 验证 □ 综合 ■ 设计 □ 创新 实验日期： 2022.11.26 实验成绩：

1. **实验目的**
2. 掌握二叉树线索化的建立
3. 掌握二叉树线索化的遍历
4. 掌握二叉树线索化的销毁
5. **实验内容**

**具体请完成如下功能：**

(1) 设计算法实现对二叉树的链式存储

(2) 中序遍历二叉树

(3) 销毁二叉树

**三、实验要求**

(1) 程序要添加适当的注释，程序的书写要采用**缩进格式**。

(2) 程序要具在一定的**健壮性**，即当输入数据非法时，程序也能适当地做出反应，如查找的内容不存在时，要给予提示等等。

(3) 程序要做到**界面友好**，在程序运行时用户可以根据相应的提示信息进行操作。

(4) 根据实验报告模板详细书写实验报告,在实验报告中给出**流程图**。

(5) **源程序和实验报告打包。二叉树的头文件夹包含文件有：BinTreeThread.h；二叉树的源程序文件夹包含文件有：BinTreeTread.c,main.c等。头文件夹、源程序文件夹和实验报告压缩为一个文件，按以下方式命名：学号姓名.rar，如0814101王五.rar。**

**四、设计实现二叉树线索化程序主要实验步骤及程序分析**

**4.1思路设计**

**4.1.1 存储结构设计**

定义两个头文件storage1.h和storage2.h，分别存储数组相关的信息：

storage1.h：

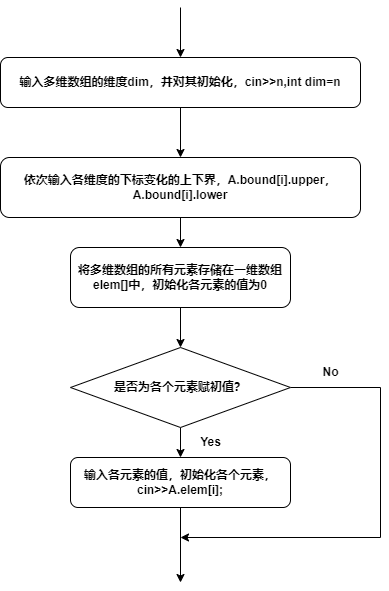
1. **typedef** **struct**
2. {
3. **int** dim,                    //数组维数
4. \*upper,                 //数组上界的指针
5. \*lower,                 //数组下界的指针
6. \*constants,            //数组映像函数常量基址
7. \*data,                  //数组元素内容
8. num;                   //数组总个数
10. }NArray;

storage2.h：

1. #include <stdarg.h>
2. #include <stdio.h>
3. #include <stdlib.h>
4. #include <malloc.h>
5. #include <conio.h>   //导入相关的头文件
6. #define MaxDim 5  //可定义数组的最大维数
7. #define maxnum 10  //定义数组的最大个数
8. #define max 100  //自定义数组中含的最大元素个数

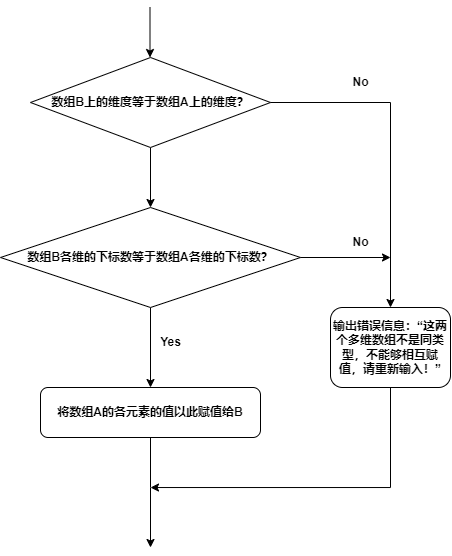
**4.1.2 算法设计**

(1) 根据所设计的存储结构对数组进行创建（初始化）， NArray creatone()， 过程如图1所示：



**图1 多维数组创建流程图**

(2) 数组赋值int qualify(NArray &A,NArray &B) ，例如，a[1..n]=a[2..n+1] 、a[2..4][3..5]=b[1..3][2..4]；要求被赋值的数组与赋值数组之间的类型（形状）要相同， 及维数相同，各维的下标变化幅度大小相等。相同类型的数组才能相互赋值，流程图如图2 所示。



**图2 多维数组赋值流程图**

**4.2主要具体步骤（完整代码见附录）**

**4.2.1 复杂度分析**

**空间复杂度：**分析该存储结构可知， 因为多维数组在物理位置上存储在一维数组中，所以对多维数组 NArray[a1…b1][a2…b2]…[an…bn]所占用的存储空间为：

(b1- a1)\*(b2 -a2)\*……\*(bn –an)。

**时间复杂度：**各个函数考虑其最主要操作，得到如下结论：

1. NArray creatone() //创建一多维数组 O(dim)
2. **int** qualify(NArray &A,NArray &B) //创建一多维数组,赋值函数 O(total)
3. **void** printit(NArray A) //打印多维数组的有关信息 O(total)
4. **void** find(NArray &A)//根据查找要求找到特定元素 O(dim\*dim)

**注： dim 为该多维数组的维数， total 为该多维数组的元素个数， 这些信息在数组的存储结构定义中均有记录**

**4.2.2 下标变量检查**

设计了一个鲁棒性强的逻辑系统，用来定义整型多维数组类型，保证各维的下标是任意整数开始的连续整数的同时，能够实现对下标变量赋值，执行下标范围检查的功能：

1. **void** InitNArray(NArray &A) {               //定义整型多维数组
2. **int** i, x, y, z, p, j, key = 1, elemtotal, address = 0;
3. printf("请输入多维数组的维数：\n");           //输入多维数组的维数
4. scanf("%d", &x);
5. **while** (x < 1 || x > MaxDim) {
6. printf("维数超出处理范围!!请重新输入\n");
7. printf("请重新输入多维数组的维数：\n");
8. scanf("%d", &x);
9. }
10. A.dim = x;
11. A.upper = (**int** \*)malloc(A.dim \* **sizeof**(**int**));        //分配空间给A.upper
12. A.lower = (**int** \*)malloc(A.dim \* **sizeof**(**int**));        //分配空间给A.lower
13. **for** (i = 1; i <= A.dim; i++) {
14. A.lower[i] = 0;                                      //给A.lower初始化为0
15. A.upper[i] = 0;                                      //给A.upper初始化为0
16. }
17. **for** (i = 1; i <= A.dim; i++) {                      //循环输入多维数组各维的上下界
18. key = 1;                                         //设置标志
19. **while** (key) {                                         //循环输入多维数组各维的上界
20. printf("请输入你要创建的%d维数组第%d维的上界：\n", A.dim, i);
21. scanf("%d", &y);
22. **while** (y < 0) {
23. printf("你输入的上界超出处理范围!请重新输入\n");
24. printf("请输入你要创建的%d维数组第%d维的上界：\n", A.dim, i);
25. scanf("%d", &y);
26. }
27. A.upper[i] = y;                                 //将各维上界赋值给A.upper[i]
28. //循环输入多维数组各维的下界
29. printf("请输入你要创建的%d维数组第%d维的下界：\n", A.dim, i);
30. scanf("%d", &z);
31. **while** (z < 0) {
32. printf("你输入的下界超出处理范围!请重新输入\n");
33. printf("请输入你要创建的%d维数组第%d维的下界：\n", A.dim, i);
34. scanf("%d", &z);
35. }
36. A.lower[i] = z;                                //将各维下界赋值给A.lower[i]
37. **if** (A.upper[i] > A.lower[i])
38. key = 0;
39. **else** {
40. printf("你所输入的下界大于上界，请按任意键返回!\n");
41. getch();
42. }
43. }
44. }

**4.2.3 确定数组的大小**

通过for循环遍历，可以输出数组的大小

1. **for** (i = 0; i <= A.dim; i++) {
2. elemtotal = elemtotal \* (A.upper[i] - A.lower[i] + 1); //计算数组总个数
3. }

**4.2.4 同类型数组赋值**

同类型赋值功能代码如下，通过构建直观的UI，可以直观地进行同类型数组赋值：

1. **void** Assign1(NArray &A) {      //对多维数组某个元素进行赋值
2. **int** i, key, address, data, a[5];
3. **for** (i = 1; i <= A.dim; i++) {
4. key = 1;
5. **while** (key) {
6. //循环输入进行赋值的元素各维下标
7. printf("请输入你要进行赋值的%d维数组中的元素第%d维下标是:", A.dim, i);
8. scanf("%d", &a[i - 1]);
9. **if** (a[i - 1] <= A.upper[i] && a[i - 1] >= A.lower[i])
10. key = 0;
11. **else** {
12. printf("输入的下标超出定义的范围，请按任意键返回再输入!\n");
13. getch();
14. }
15. }
16. }
17. //输出提示语句你要存入的位置是A[x][x]请输入你要对该元素赋值的内容
18. printf("你要存入的位置是A");
19. **for** (i = 0; i < A.dim; i++)
20. printf("[%d]", a[i]);
21. printf("请输入你要对该元素赋值的内容:\n");
22. scanf("%d", &data);
23. address = 0;      //计算地址
24. **for** (i = 1; i <= A.dim; i++) {
25. address = address + A.constants[i] \* (a[i - 1] - A.lower[i]); //地址计算公式
26. }
27. A.data[address] = data;
28. printf("              赋值成功\n");                           //输出赋值结果
29. printf("                  你对%d维数组进行赋值的结果是A", A.dim);
30. **for** (i = 0; i < A.dim; i++)
31. printf("[%d]", a[i]);
32. printf("=");
33. printf("%d\n", A.data[address]);

**4.2.5 子数组赋值**

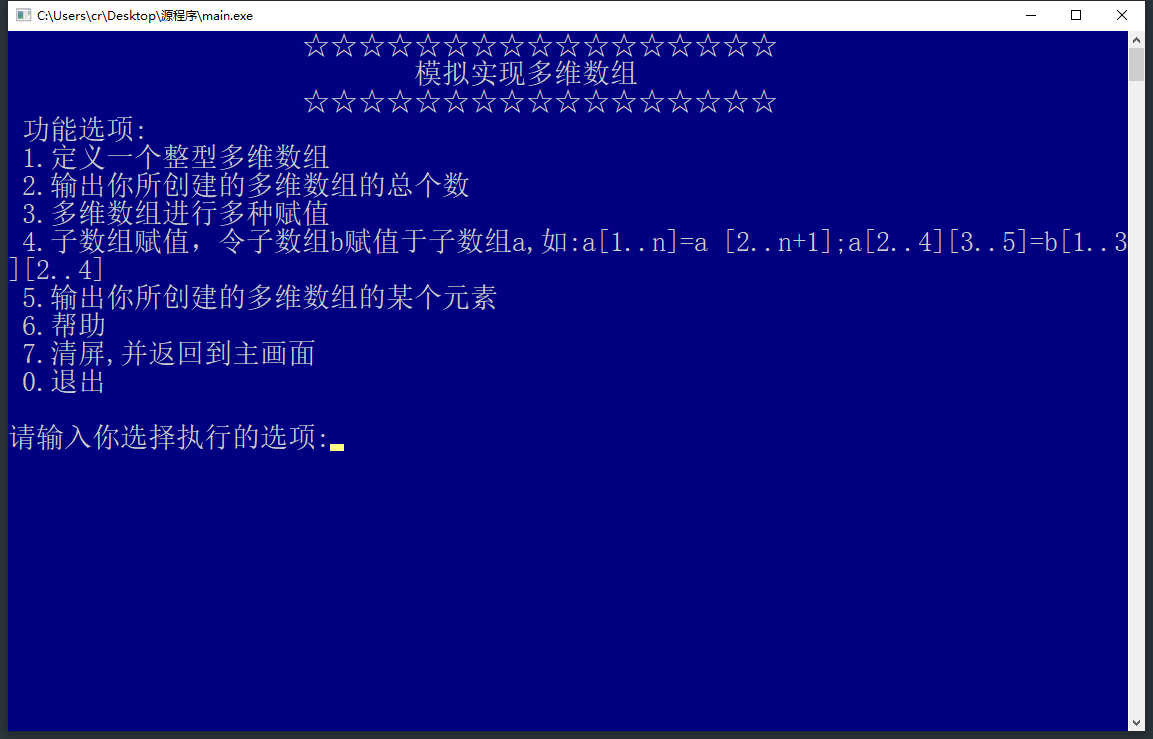
子数组赋值功能代码如下，通过构建直观的UI，可以直观地进行子数组赋值：

1. **void** Assign2 (NArray &A) {               //对多维数组中指定的子数组赋值
2. **int** i, key, b[10], address1, address2, k = 1, data; //初始化变量
3. printf("请输入子数组的第一个元素的下标:\n");
4. **for** (i = 1; i <= A.dim; i++) {
5. key = 1;
6. **while** (key) {
7. printf("子数组第一个元素的下标%d是:", i);
8. scanf("%d", &b[i - 1]);
9. **if** (b[i - 1] >= A.lower[i])
10. key = 0;
11. **else** {
12. printf("你输入的下标的结果超出定义的范围，请按任意键返回重新输入!\n");
13. getch();
14. }
15. }
16. }                                      //循环输入子数组b最后一个元素的下标
17. printf("请输入子数组的最后一个元素的下标:\n");
18. **for** (i = 1; i <= A.dim; i++) {
19. key = 1;
20. **while** (key) {
21. printf("子数组最后一个元素的下标%d是:", i);
22. scanf("%d", &b[i + 4]);
23. **if** (b[i + 4] >= A.lower[i])
24. key = 0;
25. **else** {
26. printf("你输入的下标的结果超出定义的范围，请按任意键返回重新输入!\n");
27. getch();
28. }
29. }
30. }
31. printf("你输入的子数组是A");  //输出子数组
32. **for** (i = 0; i < A.dim; i++)
33. printf("[%d]", b[i]);
34. printf("到A");
35. **for** (i = 0; i < A.dim; i++)
36. printf("[%d]", b[i + 5]);
37. printf("\n");
38. printf("请输入你要对该子数组所有元素赋值的内容:\n");
39. scanf("%d", &data);
40. address1 = 0;
41. address2 = 0;
42. **for** (i = 1; i <= A.dim; i++) {
43. address1 = address1 + A.constants[i] \* (b[i - 1] - A.lower[i]);
44. }
45. **for** (i = 1; i <= A.dim; i++) {
46. address2 = address2 + A.constants[i] \* (b[i + 4] - A.lower[i]);
47. }
48. **for** (i = address1; i <= address2; i++) {
49. A.data[i] = data;
50. }

**4.3程序结果**

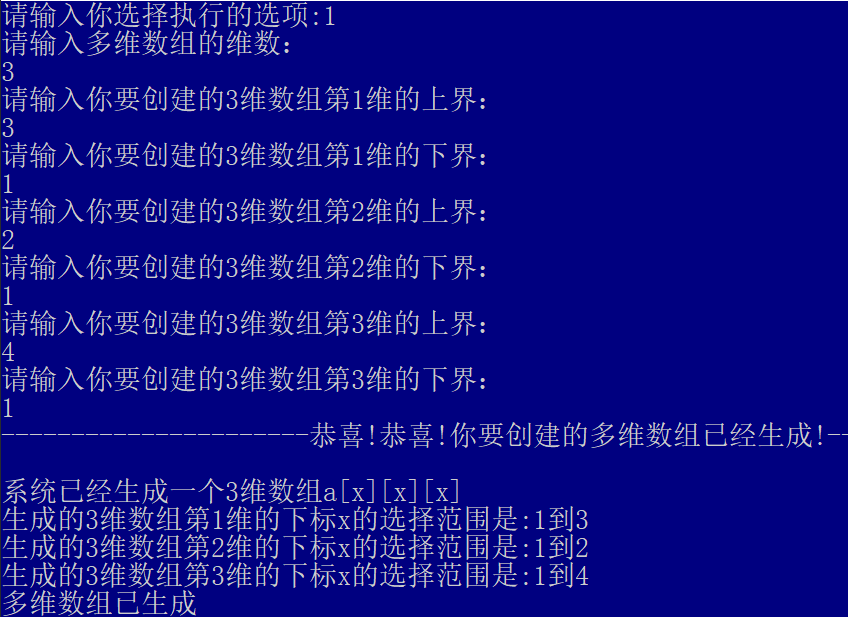
**4.3.1 UI设计**

实现了一个直观的用户操作界面，功能齐全：



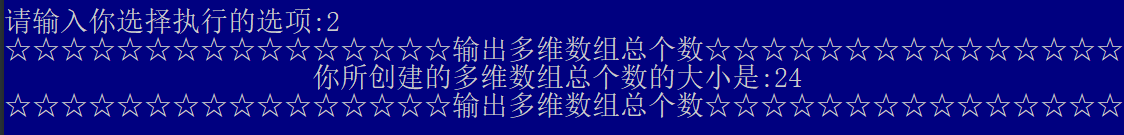
**图3 UI设计界面图**

**4.3.2 定义一个整型多维数组**



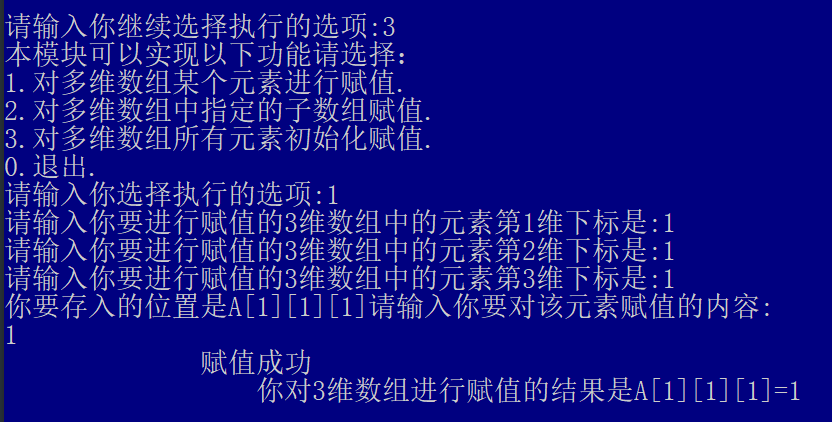
**图4 定义一个整型多维数组输出图**

**4.3.3 输出多维数组的总个数**



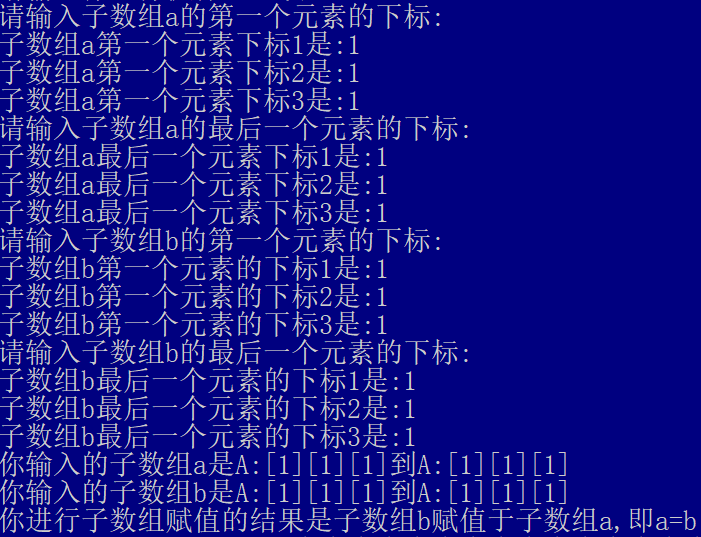
**图5 输出多维数组的总个数输出图**

**4.3.4 对多维数组的某个元素进行赋值**



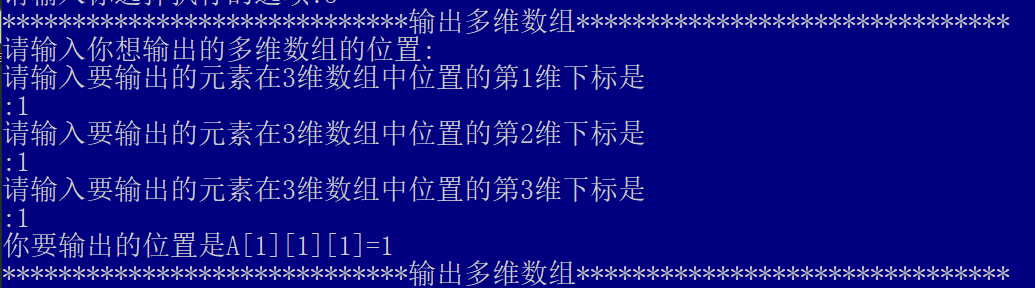
**图6 对多维数组的某个元素进行赋值输出图**

**4.3.5 对多维数组的子数组进行赋值**

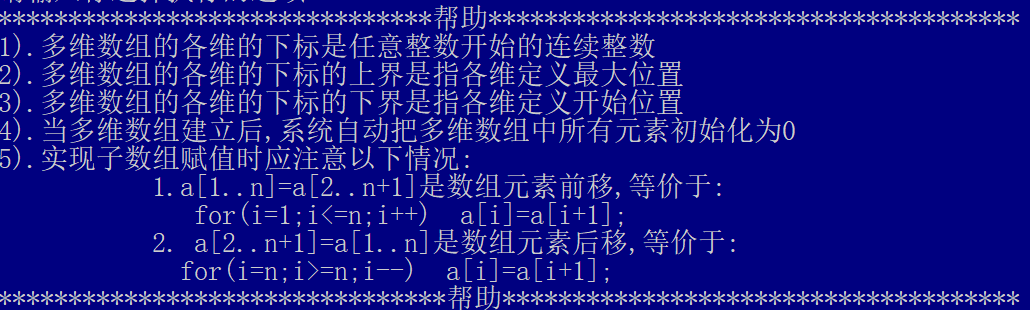


**图7 对多维数组的子数组进行赋值输出图**

**4.3.6 输出多维数组的某个元素**

**图8 输出多维数组的某个元素输出图**

**4.3.7 帮助**



**图9 帮助界面图**

**五、编写代码时遇到的问题及实验心得体会**

**遇到的问题：**

(1) 由于数组运算过程中的最重要的是下标之间的转换，一不小心很容易犯错误，在这方面，编程时犯了类似的好几个错误，影响了编程的效率，以后编程要仔细一些。

**实验心得体会：**

(1) 学会了设计比较精美的UI界面，也学习到了模块化思想解决数据结构的问题的宝贵经验，编写的程序要给人一种思路清晰的感觉， 能够与用户进行交流（在此次课程设计中，

与用户交流明显不足，这值得进一步改进），与此同时，还要注重程序的可读性、健壮

性。

(2) 老师的题目很难，也让我知道了自己的不足，在编程时候应该很仔细，对于一般的语法错误应该尽量避免。同时对于一些模棱两可的想法应该尽快查阅书本， 不能混过去， 这样很容易导致在后期的调试过程中出现大量不可预期的错误。

(3) 这次编程坚决没有上网搜索现成的方案，锻炼了自己的意志力和克服困难的能力，很感谢老师的这次考验，精通数据结构不是一朝一夕所能完成的事情，需要锲而不舍的努力。我相信我有信心与毅力将这些知识理解透彻，提高自己的综合能力。