# 数据结构学习报告——数组

学号：19040500120 姓名：吉皓哲

## 一、实验目的

1、掌握数组的基本定义、初始化

2、掌握数组的基本操作——遍历、查找等

3、掌握基于数组的基本运用——函数传参、返回值等

## 二、学习内容

1. 分别建立一个一维int和char数组，其中int数组长度为你的学号位数，char数组长度为12。分别利用你的学号和姓名对int数组和char数组做初始化后输出。
2. 2.1定义一个长度不超过30的一维int数组，程序执行时，由用户输入正整数N，然后依次输入N个整数存放在int数组里并输出获得的数组。接着输入一个整数作为关键字在int数组中查找该关键字，找到后，输出该关键字所在的位置，同时将该关键字从数组中删除（要求后续整数依次前移），随后输出删除后的新数组。

2.2定义一个长度不超过30的一维int数组，程序执行时，由用户依次输入N个整数，并根据整数的大小，将其插入到int数组中合适的位置，要求最终等到的数组为从小到大的有序数组。

1. 分别定义两个3X6和一个6X3的二维int数组，其中一个3X6矩阵在定义时完成初始化，随后通过用户输入，完成对另外一个3X6矩阵的输入与存储。接着，对该矩阵进行转置，将结果存放再6X3矩阵中。第二步对两个3X6矩阵进行加法运算，将结果存放在任意一个3X6矩阵中，并利用该结果矩阵和6X3矩阵进行乘法运算。要求每一步输出各矩阵结果。
2. 定义一个10X10的二维int数组，定义时利用初始化获得一个稀疏矩阵并输出该矩阵，随后读取该数组，利用三元组（一个一维结构体数组）储存稀疏矩阵中的非零元素（三元组分别储存非零元素的行下标，列下标和元素值）。除了三元组表本身，储存一个稀疏矩阵还需要额外的三个变量，分别储存矩阵的非零元个数，矩阵的行数和矩阵的列数。在生成三元组后，生成该矩阵的转置矩阵三元组并输出其实际矩阵阵列。随后，利用这两个矩阵，执行矩阵和操作并输出执行后的结果。
3. 给定一个一维char数组并存放若干字符（数组长度、存放进字符的方法不做限制，可以自由发挥），随后，定义一个sort函数，将刚才的char数组作为实参传递给sort函数，由后者完成对这个数组的升序排列（即结果数组由小到大排列）。要求在排序前和排序后分别输出这个数组。
4. 定义一个xorQuery函数，实现如下功能。（LeetCode No：1310子数组异或查询）

有一个正整数数组 arr，现给你一个对应的查询数组 queries，其中 queries[i] = [Li, Ri]。

对于每个查询 i，请你计算从 Li 到 Ri 的 XOR 值（即 arr[Li] xor arr[Li+1] xor ... xor arr[Ri]）作为本次查询的结果。

并返回一个包含给定查询 queries 所有结果的数组。

示例 1：

输入：arr = [1,3,4,8], queries = [[0,1],[1,2],[0,3],[3,3]]

输出：[2,7,14,8]

解释：

数组中元素的二进制表示形式是：

1 = 0001

3 = 0011

4 = 0100

8 = 1000

查询的 XOR 值为：

[0,1] = 1 xor 3 = 2

[1,2] = 3 xor 4 = 7

[0,3] = 1 xor 3 xor 4 xor 8 = 14

[3,3] = 8

示例 2：

输入：arr = [4,8,2,10], queries = [[2,3],[1,3],[0,0],[0,3]]

输出：[8,0,4,4]

## 三、算法描述

### 题目1：

1、利用char类型数组储存姓名，利用int类型数组储存学号，之后分别采用字符串类型和整数类型进行输出。

### 题目2：

2.1

1. 自然语言描述

题目中表明数组长度不超过30，故在程序开始建立长度为30的数组，用一个变量储存用户输入的正整数N，然后采用循环的方式将输入的数据储存在数组中，再将其输出。然后识别用户输入的关键词，将其储存在自由变量key中，利用循环将数组中的每一个元素与key比较，若相等，记录元素位置，并将其输出，最后再用循环遍历数组，输出与关键词不等的元素。

1. 伪代码描述

\* >>array[30]

\* >>n

\* print (array)

\* >>k

\* for ( i = 0; i < n; i++) {

\* if (k == a[i]) {

\* print (i+1)

\* }

\* }

\* for ( i = 0; i < n; i++) {

\* if (a[i] != k) {

\* printf("%d ", a[i])

\* }

\* }

2.2

1. 自然语言描述

题目中表明数组长度不超过30，故在程序开始建立长度为30的数组，然后采用while循环的形式记录用户输入，当用户输入终止符时停止记录，并为数组长度计数。采用冒泡排序算法对数据进行排序，排序完成后打印排序结果

1. 伪代码描述

\* >>array[30]

\* while ( input != EOF ){

\* num++

\* }

\* bubblesort( array )

\* output array[ num-1 ]

### 题目3：

1. 自然语言描述

建立两个3×6矩阵（矩阵a、b），一个6×3矩阵（矩阵c），一个3×3矩阵（矩阵d），在程序开头完成对一个3×6矩阵的初始化，利用双重for循环实现第二个3×6矩阵的用户输入。矩阵转置时，利用双重for循环，将b矩阵第i行第j列的元素赋给c矩阵第j行第i列的元素即可。矩阵求和时，利用双重for循环，将b矩阵第i行第j列的元素与a矩阵第i行第j列的元素相加赋给b矩阵第i行第j列的元素即可。矩阵求积时应用i×s与s×j矩阵求积公式，构造三重for循环求积，并将求积结果赋值给矩阵d

1. 伪代码描述

\* >>a[3][6], b[3][6], c[6][3], d[3][3]

\* for ( i = 0; i < 3; i++) {

\* for ( j = 0; j < 6; j++) {

\* input b[i][j]

\* }

\* }

\* //矩阵转置

\* for ( i = 0; i < 3; i++) {

\* for ( j = 0; j < 6; j++) {

\* c[j][i] = b[i][j]

\* }

\* }

\* output c

\* //用b矩阵储存和矩阵

\* for (int i = 0; i < 3; i++) {

\* for (int j = 0; j < 6; j++) {

\* b[i][j] = b[i][j] + a[i][j]

\* }

\* }

\* output b

\* //矩阵求积

\* for ( i = 0; i < 3; i++) {

\* for ( j = 0; j < 3; j++) {

\* d[i][j] = 0;

\* for ( k = 0; k < 6; k++) {

\* temp = b[i][k] \* c[k][j];

\* d[i][j] = d[i][j] + temp;

\* }

\* }

\* }

\* output d

### 题目4

1. 自然语言描述

建立两个10×10的二位数组用来储存稀疏矩阵及其和矩阵，建立一个包含三个int变量的结构体，分别储存稀疏矩阵中非零元素的横纵坐标和值，再建立一个以该结构体为元素的数组，利用二重for循环遍历稀疏矩阵，找出非零元素的位置和值，获得该稀疏矩阵的三元组，并打印三元组，在利用与题3相同的方法将该系数矩阵转置，再次遍历获得转置矩阵三元组。

1. 伪代码描述

\* >>a[10][10],c[10][10]

\* struct Triele

\* {

\* int i;

\* int j;

\* int value;

\* }

\* >> Triele b[5],d[5]

\* //获得稀疏矩阵三元组

\* for (int x = 0; x < 10; x++) {

\* for (int y = 0; y < 10; y++) {

\* if (a[x][y] != 0) {

\* b[cnt].i = x + 1;

\* b[cnt].j = y + 1;

\* b[cnt].value = a[x][y];

\* cnt++;

\* }

\* }

\* }

\* Output b[5]

\* //矩阵转置

\* for ( x = 0; x < 10; x++) {

\* for ( y = 0; y < 10; y++) {

\* c[y][x] = a[x][y];

\* }

\* }

\* Get d[5]

\* Output d[5]

\* Output c[10][10]

### 题目5

1. 自然语言描述

此题与2.2题要求类似，只需要将2.2题使用的排序算法转化为函数，在排序的比较过程中将字符类型强制转换为int型即可。

1. 伪代码描述

\* >>c[10]

\* Output c

\* //排序函数

\* sort( c )

\* Output c

### 题目6

1. 自然语言描述

题目要求我们生成查询结果的数组，首先应该考虑到当正整数数组arr或者查询数组queries为空时，得不到任何查询结果，应当返回null。根据题目要求，The returned array must be malloced, assume caller calls free()，我们应该用malloc来定义返回值数组。

在查询时，考虑使用prefix sum来计算XOR，定义pre数组，利用循环结构进行迭代，pre[i]的值为第一个元素到第i+1个元素的XOR值。

当进行查询时，首先应当考虑每个查询的结构，当查询的起始值begin和终止值end相同时，则该查询仅包含一个元素，查询值即为该位置的arr数组元素值，当查询的起始值begin和终止值end不同时，若查询的起始值begin为0，则查询值与终止值end对应的prefix sum相等；若查询的起始值begin不为0，则利用XOR运算的x ^ y ^ x = y的性质，我们可以用begin – 1对应的prefix sum与end对应的prefix sum取XOR运算，即可得到查询值。然后将生成的查询值数组ans返回给主函数即可。

## 四、程序源码

### 题目1

#include <stdio.h>

int main(void)

{

char a[12] = { "吉皓哲" }; //用char类型数组储存姓名

int b[11] = { 1 , 9 , 0 , 4 , 0 , 5 , 0 , 0 , 1 , 2 , 0 }; //用int类型数组储存学号

//采用遍历的方式输出学号

for (int i = 0; i < 11; i++) {

printf("%d", b[i]);

}

printf("\n");

printf("%s\n", a); //输出字符串

return 0;

}

### 题目2.1

#include <stdio.h>

int main(void) {

int a[30];

int n; //用户输入的正整数N

scanf("%d", &n);

//利用for循环将用户输入的数据存在数组中

for (int i = 0; i < n; i++) {

scanf("%d", &a[i]);

}

//利用for循环将用户输入的数据输出

for (int i = 0; i < n; i++) {

printf("%d ", a[i]);

}

printf("\n");

int k; //用户输入的关键词

scanf("%d", &k);

//利用for循环将关键词与数组比较

for (int i = 0; i < n; i++) {

if (k == a[i]) {

printf("%d ", i + 1);

}

}

printf("\n");

//利用for循环输出数组

for (int i = 0; i < n; i++) {

if (a[i] != k) { //如果数组中的数据与关键词相同，则不输出

printf("%d ", a[i]);

}

}

return 0;

}

### 题目2.2

#include <stdio.h>

int main(void) {

int arr[30]; //定义一个长度不超过30的一维int数组

int num = 0; //num为数组下标

//利用for循环将用户输入的数据储存在数组中

while (scanf\_s("%d", &arr[num]) != EOF) { //CTRL+Z表示输入结束(VS需要连续输入三次CTRL+Z)

num++;

}

int i, j, temp = 0; //i为排序次数控制变量，j为数组下标，

//temp为交换过程中使用的临时变量

//冒泡排序

for (i = 0; i < num - 1; i++) { //从后向前进行比较，每次比较完成，参与比较的数中最小的一个被放在最前端

for (j = num - 1; j > i; j--) { //每次排序完成后，需要参与排序的数少了一个，循环次数减少一次

if (arr[j] < arr[j - 1]) { //将大的数字放在后面

temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

}

}

}

//利用for循环输出排序后的数组

for (int i = 0; i < num; i++) {

printf("%d ", arr[i]);

}

return 0;

}

### 题目3

#include <stdio.h>

int main(void) {

int a[3][6] =

{

{ 1, 0, 0, 0, 0, 0 },

{ 0, 1, 0, 0, 0, 0 },

{ 0, 0, 1, 0, 0, 0 }

}; //建立矩阵a，并对a进行初始化

int b[3][6];

int c[6][3]; //用来储存转置后的矩阵

int d[3][3]; //建立6×6矩阵用来存放相乘后的矩阵

//用户输入第二个3×6矩阵

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 6; j++) {

scanf\_s("%d", &b[i][j]);

}

}

//矩阵转置

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 6; j++) {

c[j][i] = b[i][j];

}

}

//输出转置后的矩阵

printf("这是转置后的矩阵\n");

for (int j = 0; j < 6; j++) {

for (int i = 0; i < 3; i++) {

printf("%d ",c[j][i]);

}

printf("\n");

}

//用b矩阵储存和矩阵

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 6; j++) {

b[i][j] = b[i][j] + a[i][j];

}

}

//输出和矩阵

printf("这是和矩阵\n");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 6; j++) {

printf("%d ", b[i][j]);

}

printf("\n");

}

//矩阵求积

int temp; //储存元素结果的临时变量

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

d[i][j] = 0;

for (int k = 0; k < 6; k++) {

temp = b[i][k] \* c[k][j]; //d(i,j) = segma(k = 1 → 6)(a(i,k)×b(k,j)

d[i][j] = d[i][j] + temp;

}

}

}

//输出相乘后的矩阵

printf("这是积矩阵\n");

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

printf("%d ", d[i][j]);

}

printf("\n");

}

return 0;

}

### 题目4

#include <stdio.h>

int main(void) {

//初始化稀疏矩阵

int a[10][10] =

{

{ 1 },

{ 0, 2 },

{ 0, 0, 1 },

{ 3 },

{ 1 }

};

int c[10][10]; //用于储存转置矩阵

//打印初始化得到的稀疏矩阵

printf("初始化的稀疏矩阵为：\n");

for (int x = 0; x < 10; x++) {

for (int y = 0; y < 10; y++) {

printf("%d ", a[x][y]);

}

printf("\n");

}

//建立三元组结构体

struct Triele

{

int i;

int j;

int value;

};

//建立稀疏矩阵三元组数组

Triele b[5];

//获得稀疏矩阵三元组

int cnt = 0;

for (int x = 0; x < 10; x++) {

for (int y = 0; y < 10; y++) {

if (a[x][y] != 0) {

b[cnt].i = x + 1;

b[cnt].j = y + 1;

b[cnt].value = a[x][y];

cnt++;

}

}

}

//打印稀疏矩阵三元组

printf("该矩阵的三元组为：\n");

for (int x = 0; x < 5; x++) {

printf("%d %d %d\n", b[x].i, b[x].j, b[x].value);

}

//矩阵转置

for (int x = 0; x < 10; x++) {

for (int y = 0; y < 10; y++) {

c[y][x] = a[x][y];

}

}

//建立转置矩阵三元组数组

Triele d[5];

//获得转置矩阵三元组

cnt = 0;

for (int x = 0; x < 10; x++) {

for (int y = 0; y < 10; y++) {

if (c[x][y] != 0) {

d[cnt].i = x + 1;

d[cnt].j = y + 1;

d[cnt].value = c[x][y];

cnt++;

}

}

}

//打印转置矩阵三元组

printf("该转置矩阵的三元组为：\n");

for (int x = 0; x < 5; x++) {

printf("%d %d %d\n", d[x].i, d[x].j, d[x].value);

}

//打印转置矩阵

printf("转置矩阵为：\n");

for (int x = 0; x < 10; x++) {

for (int y = 0; y < 10; y++) {

printf("%d ", c[x][y]);

}

printf("\n");

}

//矩阵求和

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

c[i][j] = c[i][j] + a[i][j];

}

}

//输出和矩阵

printf("这是和矩阵：\n");

for (int i = 0; i < 10; i++) {

for (int j = 0; j < 10; j++) {

printf("%d ", c[i][j]);

}

printf("\n");

}

return 0;

}

### 题目5

#include <stdio.h>

//sort函数声明

void sort(char\*, int);

int main(void) {

//数组定义

char c[10] = { 'a','A','b','B','c','C','d','D','e','E' };

//打印排序前的数组

printf("排序前的数组为：");

for (int i = 0; i < 10; i++) {

printf("%c ", c[i]);

}

printf("\n");

sort(c, 10); //排序函数

//打印排序后的数组

printf("排序后的数组为：");

for (int i = 0; i < 10; i++) {

printf("%c ", c[i]);

}

return 0;

}

//sort函数定义

void sort(char\* c, int n) {

int temp;

for (int i = 0; i < n - 1 ; i++) {

for (int j = n - 1; j > i; j--) { //每次排序完成后，参与排序的最小数已被放置在数组最前端，循环次数减少一次

if (int(c[j]) < int(c[j - 1])) { //将字符类型强制转换为int类型，将大的数字放在后面

temp = c[j];

c[j] = c[j - 1];

c[j - 1] = temp; //从后向前进行比较，每次比较完成，参与比较的数中最小的一个被放在最前端

}

}

}

}

### 题目6

int\* xorQueries(int\* arr, int arrSize, int\*\* queries, int queriesSize, int\* queriesColSize, int\* returnSize) {

if (arrSize == 0 || queriesSize == 0) {

return NULL; //如果数组为空或查询数组为空，则返回为NULL

}

int pre[arrSize + 1];

int\* ans = (int\*)malloc(sizeof(int) \* queriesSize);

pre[0] = arr[0];

for (int i = 1; i < arrSize; i++) {

pre[i] = pre[i - 1] ^ arr[i]; //利用prefix进行前缀和的计算

// printf("pri[%d]:%d\n", i, pri[i]);

}

for (int i = 0; i < queriesSize; i++) {

int begin = queries[i][0];

int end = queries[i][1];

if (begin != end) {

if (begin == 0) {

ans[i] = pre[end]; //如果从数组的第一个元素开始查询，则该查询值与前缀和相等

}

else {

ans[i] = pre[begin - 1] ^ pre[end]; //利用x ^ y ^ x = y的性质

}

}

else {

ans[i] = arr[end]; //如果查询数组仅包含一个元素，则返回该元素的值

}

// printf("[%d]:%d\n", i, ans[i]);

}

\*returnSize = queriesSize;

return ans;

}

## 五、学习报告和心得体会

### 关于冒泡排序算法的调试过程

在进行数据排序的过程中首先通过网络查找，得到如下代码思路：

for (i = 0; i < n; i++){ //将数组遍历n-1次

for (j = 1; j < n - i; j++){

if (s[j] < s[j - 1]){ //每次遍历，比较相邻两数大小

t = s[j]; //若后面的数字小，则交换

s[j] = s[j - 1];

s[j - 1] = t;

}

}

}

该算法属于冒泡排序常规算法，查阅相关资料后，通过动画图解，了解到一种改进的算法，大体思路与概述如下：

将两数的比较视为一个天平的称量，将天平从数组的尾端移至前端，第一次移动可以保证数组中最小的数被放在了数组最前端。这时将天平移回数组尾端，重新开始向前移动号，但现在最前端的一个数可以不用参与排序。

代码概述如下：

for (i = 0; i < num - 1; i++) { //从后向前进行比较，每次比较完成，参与比较的数中最小的一个被放在最前端

for (j = num - 1; j > i; j--) { //每次排序完成后，需要参与排序的数少了一个，循环次数减少一次

if (arr[j] < arr[j - 1]) { //将大的数字放在后面

temp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = temp;

}

}

}

该算法每次遍历过后，即可将当前参与排序的数字中最小的一个移至数组最前端，下一次遍历时就可以少遍历一个数据。